```
**********************************************************
   13892 Wed Jan 31 21:21:15 2007
new/usr/src/common/zfs/zfs_prop.c
1000002 gzip compression for ZFS
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "@(#)zfs_prop.c 1.17    07/01/31 SMI"
#pragma ident   "@(#)zfs_prop.c 1.16    07/01/05 SMI"

/*
 * Master property table.
 *
 * This table keeps track of all the properties supported by ZFS, and their
 * various attributes.  Not all of these are needed by the kernel, and several
 * are only used by a single libzfs client.  But having them here centralizes
 * all property information in one location.
 *
 *      name            The human-readable string representing this property
 *      proptype        Basic type (string, boolean, number)
 *      default         Default value for the property.  Sadly, C only allows
 *                      you to initialize the first member of a union, so we
 *                      have two default members for each property.
 *      attr            Attributes (readonly, inheritable) for the property
 *      types           Valid dataset types to which this applies
 *      values          String describing acceptable values for the property
 *      colname         The column header for 'zfs list'
 *      colfmt          The column formatting for 'zfs list'
 *
 * This table must match the order of property types in libzfs.h.
 */

#include <sys/zio.h>
#include <sys/spa.h>
#include <sys/zfs_acl.h>
#include <sys/zfs_ioctl.h>

#include "zfs_prop.h"

#if defined(_KERNEL)
#include <sys/systm.h>
#else
#include <stdlib.h>
```



```
#include <string.h>
#include <ctype.h>
#endif

typedef enum {
        prop_default,
        prop_readonly,
        prop_inherit
} prop_attr_t;
```

*_____unchanged_portion_omitted_*

```
static prop_desc_t zfs_prop_table[ZFS_NPROP_ALL] = {
        { "type",       prop_type_string,       0,      NULL,   prop_readonly,
          ZFS_TYPE_ANY, "filesystem | volume | snapshot", "TYPE", B_TRUE },
        { "creation",   prop_type_number,       0,      NULL,   prop_readonly,
          ZFS_TYPE_ANY, "<date>", "CREATION", B_FALSE },
        { "used",       prop_type_number,       0,      NULL,   prop_readonly,
          ZFS_TYPE_ANY, "<size>",     "USED", B_TRUE },
        { "available",  prop_type_number,       0,      NULL,   prop_readonly,
          ZFS_TYPE_FILESYSTEM | ZFS_TYPE_VOLUME, "<size>", "AVAIL", B_TRUE },
        { "referenced", prop_type_number,       0,      NULL,   prop_readonly,
          ZFS_TYPE_ANY,
          "<size>", "REFER", B_TRUE },
        { "compressratio", prop_type_number,    0,      NULL,   prop_readonly,
          ZFS_TYPE_ANY, "<1.00x or higher if compressed>", "RATIO", B_TRUE },
        { "mounted",    prop_type_boolean,      0,      NULL,   prop_readonly,
          ZFS_TYPE_FILESYSTEM, "yes | no | -", "MOUNTED", B_TRUE },
        { "origin",     prop_type_string,       0,      NULL,   prop_readonly,
          ZFS_TYPE_FILESYSTEM | ZFS_TYPE_VOLUME, "<snapshot>", "ORIGIN",
          B_FALSE },
        { "quota",      prop_type_number,       0,      NULL,   prop_default,
          ZFS_TYPE_FILESYSTEM, "<size> | none", "QUOTA", B_TRUE },
        { "reservation", prop_type_number,      0,      NULL,   prop_default,
          ZFS_TYPE_FILESYSTEM | ZFS_TYPE_VOLUME,
          "<size> | none", "RESERV", B_TRUE },
        { "volsize",    prop_type_number,       0,      NULL,   prop_default,
          ZFS_TYPE_VOLUME, "<size>", "VOLSIZE", B_TRUE },
        { "volblocksize", prop_type_number,     8192,   NULL,   prop_readonly,
          ZFS_TYPE_VOLUME, "512 to 128k, power of 2", "VOLBLOCK", B_TRUE },
        { "recordsize", prop_type_number,       SPA_MAXBLOCKSIZE,       NULL,
          prop_inherit,
          ZFS_TYPE_FILESYSTEM,
          "512 to 128k, power of 2", "RECSIZE", B_TRUE },
        { "mountpoint", prop_type_string,       0,      "/",    prop_inherit,
          ZFS_TYPE_FILESYSTEM,
          "<path> | legacy | none", "MOUNTPOINT", B_FALSE },
        { "sharenfs",   prop_type_string,       0,      "off",  prop_inherit,
          ZFS_TYPE_FILESYSTEM,
          "on | off | share(1M) options", "SHARENFS", B_FALSE },
        { "shareiscsi", prop_type_string,       0,      "off",  prop_inherit,
          ZFS_TYPE_ANY,
          "on | off | type=<type>", "SHAREISCSI", B_FALSE },
        { "checksum",   prop_type_index,        ZIO_CHECKSUM_DEFAULT,   "on",
          prop_inherit,       ZFS_TYPE_FILESYSTEM | ZFS_TYPE_VOLUME,
          "on | off | fletcher2 | fletcher4 | sha256", "CHECKSUM", B_TRUE },
        { "compression", prop_type_index,       ZIO_COMPRESS_DEFAULT,   "off",
          prop_inherit,       ZFS_TYPE_FILESYSTEM | ZFS_TYPE_VOLUME,
          "on | off | lzjb | gzip", "COMPRESS", B_TRUE },
          "on | off | lzjb", "COMPRESS", B_TRUE },
        { "atime",      prop_type_boolean,      1,      NULL,   prop_inherit,
          ZFS_TYPE_FILESYSTEM,
          "on | off", "ATIME", B_TRUE },
        { "devices",    prop_type_boolean,      1,      NULL,   prop_inherit,
          ZFS_TYPE_FILESYSTEM | ZFS_TYPE_SNAPSHOT,
          "on | off", "DEVICES", B_TRUE },
        { "exec",       prop_type_boolean,      1,      NULL,   prop_inherit,
```

```
137             ZFS_TYPE_FILESYSTEM | ZFS_TYPE_SNAPSHOT,
138             "on | off", "EXEC", B_TRUE },
139         { "setuid",     prop_type_boolean,      1,      NULL,   prop_inherit,
140             ZFS_TYPE_FILESYSTEM | ZFS_TYPE_SNAPSHOT, "on | off", "SETUID",
141             B_TRUE },
142         { "readonly",   prop_type_boolean,      0,      NULL,   prop_inherit,
143             ZFS_TYPE_FILESYSTEM | ZFS_TYPE_VOLUME,
144             "on | off", "RDONLY", B_TRUE },
145         { "zoned",      prop_type_boolean,      0,      NULL,   prop_inherit,
146             ZFS_TYPE_FILESYSTEM,
147             "on | off", "ZONED", B_TRUE },
148         { "snapdir",    prop_type_index,        ZFS_SNAPDIR_HIDDEN, "hidden",
149             prop_inherit,
150             ZFS_TYPE_FILESYSTEM,
151             "hidden | visible", "SNAPDIR", B_TRUE },
152         { "aclmode", prop_type_index,   ZFS_ACL_GROUPMASK, "groupmask",
153             prop_inherit, ZFS_TYPE_FILESYSTEM,
154             "discard | groupmask | passthrough", "ACLMODE", B_TRUE },
155         { "aclinherit", prop_type_index,        ZFS_ACL_SECURE, "secure",
156             prop_inherit, ZFS_TYPE_FILESYSTEM,
157             "discard | noallow | secure | passthrough", "ACLINHERIT", B_TRUE },
158         { "canmount",   prop_type_boolean,      1,      NULL,   prop_default,
159             ZFS_TYPE_FILESYSTEM,
160             "on | off", "CANMOUNT", B_TRUE },
161         { "xattr",      prop_type_boolean,      1,      NULL,   prop_inherit,
162             ZFS_TYPE_FILESYSTEM | ZFS_TYPE_SNAPSHOT,
163             "on | off", "XATTR", B_TRUE },
164         { "createtxg",  prop_type_number,       0,      NULL,   prop_readonly,
165             ZFS_TYPE_ANY, NULL, NULL, B_FALSE},
166         { "name",       prop_type_string,       0,      NULL,   prop_readonly,
167             ZFS_TYPE_ANY, NULL, "NAME", B_FALSE },
168         { "iscsioptions", prop_type_string,     0,      NULL,   prop_inherit,
169             ZFS_TYPE_VOLUME, NULL, "ISCSIOPTIONS", B_FALSE },
170         { "numclones", prop_type_number,        0,      NULL,   prop_readonly,
171             ZFS_TYPE_SNAPSHOT, NULL, NULL, B_FALSE },
172 };
_____unchanged_portion_omitted_

321 static zfs_index_t compress_table[] = {
322         { "on",         ZIO_COMPRESS_ON },
323         { "off",        ZIO_COMPRESS_OFF },
324         { "lzjb",       ZIO_COMPRESS_LZJB },
325         { "gzip",       ZIO_COMPRESS_GZIP },
326         { NULL }
327 };
_____unchanged_portion_omitted_
```

```
**********************************************************
    2269 Wed Jan 31 21:21:15 2007
new/usr/src/lib/libzpool/Makefile.com
1000002 gzip compression for ZFS
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)Makefile.com       1.4     07/01/31 SMI"
# ident "@(#)Makefile.com       1.3     06/08/03 SMI"
#

LIBRARY= libzpool.a
VERS= .1

# include the list of ZFS sources
include ../../../uts/common/Makefile.files
KERNEL_OBJS = kernel.o taskq.o util.o
LIST_OBJS = list.o

OBJECTS=$(ZFS_COMMON_OBJS) $(ZFS_SHARED_OBJS) $(KERNEL_OBJS) $(LIST_OBJS)

# include library definitions
include ../../Makefile.lib

ZFS_COMMON_SRCS=        $(ZFS_COMMON_OBJS:%.o=../../../uts/common/fs/zfs/%.c)
ZFS_SHARED_SRCS=        $(ZFS_SHARED_OBJS:%.o=../../../common/zfs/%.c)
KERNEL_SRCS=            $(KERNEL_OBJS:%.o=../common/%.c)
LIST_SRCS=              $(LIST_OBJS:%.o=../../../uts/common/os/%.c)

SRCS=$(ZFS_COMMON_SRCS) $(ZFS_SHARED_SRCS) $(KERNEL_SRCS) $(LIST_SRCS)
SRCDIR=         ../common

# There should be a mapfile here
MAPFILES =

LIBS +=         $(LINTLIB)

INCS += -I../common
INCS += -I../../../uts/common/fs/zfs
INCS += -I../../../common/zfs

$(LINTLIB) := SRCS=     $(SRCDIR)/$(LINTSRC)
```



```
C99MODE=        -xc99=%all
C99LMODE=       -Xc99=%all

CFLAGS +=       -g $(CCVERBOSE) $(CNOGLOBAL)
CFLAGS64 +=     -g $(CCVERBOSE) $(CNOGLOBAL)
LDLIBS +=       -lumem -lavl -lnvpair -lz -lc
LDLIBS +=       -lumem -lavl -lnvpair -lc
CPPFLAGS +=     $(INCS)

.KEEP_STATE:

all: $(LIBS)

lint: $(LINTLIB)

include ../../Makefile.targ

pics/%.o: ../../../uts/common/fs/zfs/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: ../../../common/zfs/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: ../../../uts/common/os/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)
```

```
**********************************************************
   16863 Wed Jan 31 21:21:15 2007
new/usr/src/lib/libzpool/common/kernel.c
1000002 gzip compression for ZFS
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "@(#)kernel.c   1.6     07/01/31 SMI"
#pragma ident   "@(#)kernel.c   1.5     07/01/08 SMI"

#include <assert.h>
#include <fcntl.h>
#include <sys/zfs_context.h>
#include <poll.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <zlib.h>
#include <sys/spa.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/spa.h>
#include <sys/processor.h>
#include <sys/zfs_context.h>
#include <sys/zmod.h>


/*
 * Emulation of kernel services in userland.
 */

uint64_t physmem;
vnode_t *rootdir = (vnode_t *)0xabcd1234;

/*
 * =========================================================================
 * threads
 * =========================================================================
 */
/*ARGSUSED*/
kthread_t *
zk_thread_create(void (*func)(), void *arg)
```



```
{
        thread_t tid;

        VERIFY(thr_create(0, 0, (void *(*)(void *))func, arg, THR_DETACHED,
            &tid) == 0);

        return ((void *)(uintptr_t)tid);
}
```

*unchanged_portion_omitted_*

```
void
kernel_fini(void)
{
        spa_fini();
}

int
z_uncompress(void *dst, size_t *dstlen, const void *src, size_t srclen)
{
        z_stream zs;
        int err;

        bzero(&zs, sizeof (zs));
        zs.next_in = (uchar_t *)src;
        zs.avail_in = srclen;
        zs.next_out = dst;
        zs.avail_out = *dstlen;

        if ((err = inflateInit(&zs)) != Z_OK)
                return (err);

        if ((err = inflate(&zs, Z_FINISH)) != Z_STREAM_END) {
                (void) inflateEnd(&zs);
                return (err == Z_OK ? Z_BUF_ERROR : err);
        }

        *dstlen = zs.total_out;
        return (inflateEnd(&zs));
}

int
z_compress(void *dst, size_t *dstlen, const void *src, size_t srclen)
{

        z_stream zs;
        int err;

        bzero(&zs, sizeof (zs));
        zs.next_in = (uchar_t *)src;
        zs.avail_in = srclen;
        zs.next_out = dst;
        zs.avail_out = *dstlen;

        if ((err = deflateInit(&zs, Z_DEFAULT_COMPRESSION)) != Z_OK)
                return (err);

        if ((err = deflate(&zs, Z_FINISH)) != Z_STREAM_END) {
                (void) deflateEnd(&zs);
                return (err == Z_OK ? Z_BUF_ERROR : err);
        }

        *dstlen = zs.total_out;
        return (deflateEnd(&zs));
}
```

*unchanged_portion_omitted_*

```
**********************************************************
   27268 Wed Jan 31 21:21:16 2007
new/usr/src/uts/common/Makefile.files
1000002 gzip compression for ZFS
1000003 dboot left behind some vestiges of the once proud zmod society
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)Makefile.files     1.507   07/01/31 SMI"
# ident "@(#)Makefile.files     1.506   07/01/21 SMI"
#
# This Makefile defines all file modules for the directory uts/common
# and its children. These are the source files which may be considered
# common to all SunOS systems.

i386_CORE_OBJS += \
                avintr.o        \
                pic.o

sparc_CORE_OBJS +=

COMMON_CORE_OBJS +=             \
                atomic.o        \
                bitset.o        \
                bp_map.o        \
                brand.o         \
                cmt.o           \
                cpu.o           \
                cpu_intr.o      \
                cpupart.o       \
                disp.o          \
                group.o         \
                kstat_fr.o      \
                lgrp.o          \
                lgrp_topo.o     \
                mutex.o         \
                page_lock.o     \
                page_retire.o   \
                panic.o         \
                param.o         \
                pg.o            \
                pghw.o          \
                putnext.o       \
```



```
                rctl_proc.o     \
                rwlock.o        \
                seg_kmem.o      \
                string.o        \
                thread_intr.o   \
                vm_page.o       \
                vm_pagelist.o

CORE_OBJS +=    $(COMMON_CORE_OBJS) $($(MACH)_CORE_OBJS)

GENUNIX_OBJS += \
                access.o        \
                acl.o           \
                acl_common.o    \
                adjtime.o       \
                alarm.o         \
                aio_subr.o      \
                auditsys.o      \
                autoconf.o      \
                avl.o           \
                bdev_dsort.o    \
                bio.o           \
                bitmap.o        \
                blabel.o        \
                brandsys.o      \
                callb.o         \
                callout.o       \
                chdir.o         \
                chmod.o         \
                chown.o         \
                cladm.o         \
                class.o         \
                clock.o         \
                clock_highres.o \
                clock_realtime.o\
                close.o         \
                compress.o      \
                condvar.o       \
                conf.o          \
                console.o       \
                contract.o      \
                copyops.o       \
                core.o          \
                corectl.o       \
                cred.o          \
                cs_stubs.o      \
                dacf.o          \
                dacf_clnt.o     \
                cyclic.o        \
                ddi.o           \
                ddifm.o         \
                ddi_intr.o      \
                ddi_intr_impl.o \
                ddi_nodeid.o    \
                ddi_strtol.o    \
                devcfg.o        \
                devcache.o      \
                devid.o         \
                devid_cache.o   \
                devid_scsi.o    \
                devpolicy.o     \
                disp_lock.o     \
                dnlc.o          \
                driver.o        \
                dumpsubr.o      \
                driver_lyr.o    \
```

```
126                 dtrace_subr.o   \
127                 errorq.o        \
128                 etheraddr.o     \
129                 evchannels.o    \
130                 exacct.o        \
131                 exacct_core.o   \
132                 exec.o          \
133                 exit.o          \
134                 fbio.o          \
135                 fcntl.o         \
136                 fdbuffer.o      \
137                 fdsync.o        \
138                 fem.o           \
139                 ffs.o           \
140                 fio.o           \
141                 flock.o         \
142                 fm.o            \
143                 fork.o          \
144                 vpm.o           \
145                 fsat.o          \
146                 fs_subr.o       \
147                 fsflush.o       \
148                 ftrace.o        \
149                 getcwd.o        \
150                 getdents.o      \
151                 getloadavg.o    \
152                 getpagesizes.o  \
153                 getpid.o        \
154                 gfs.o           \
155                 rusagesys.o     \
156                 gid.o           \
157                 groups.o        \
158                 grow.o          \
159                 hat.o           \
160                 hat_refmod.o    \
161                 id32.o          \
162                 id_space.o      \
163                 inet_ntop.o     \
164                 instance.o      \
165                 ioctl.o         \
166                 issetugid.o     \
167                 ippconf.o       \
168                 kdi.o           \
169                 kcpc.o          \
170                 kmem.o          \
171                 ksyms_snapshot.o        \
172                 l_strplumb.o    \
173                 labelsys.o      \
174                 link.o          \
175                 list.o          \
176                 lockstat_subr.o \
177                 log_sysevent.o  \
178                 logsubr.o       \
179                 lookup.o        \
180                 lseek.o         \
181                 lwp.o           \
182                 lwp_create.o    \
183                 lwp_info.o      \
184                 lwp_self.o      \
185                 lwp_sobj.o      \
186                 lwp_timer.o     \
187                 lwpsys.o        \
188                 main.o          \
189                 memcntl.o       \
190                 memstr.o        \
191                 lgrpsys.o       \
```



```
192                 mkdir.o         \
193                 mknod.o         \
194                 mount.o         \
195                 move.o          \
196                 msacct.o        \
197                 multidata.o     \
198                 nbmlock.o       \
199                 ndifm.o         \
200                 nice.o          \
201                 netstack.o      \
202                 ntptime.o       \
203                 nvpair.o        \
204                 nvpair_alloc_system.o   \
205                 nvpair_alloc_fixed.o    \
206                 open.o          \
207                 p_online.o      \
208                 pathconf.o      \
209                 pathname.o      \
210                 pause.o         \
211                 serializer.o    \
212                 pci_intr_lib.o  \
213                 pci_cap.o       \
214                 pcifm.o         \
215                 pgrp.o          \
216                 pgrpsys.o       \
217                 pid.o           \
218                 policy.o        \
219                 poll.o          \
220                 pool.o          \
221                 pool_pset.o     \
222                 port_subr.o     \
223                 ppriv.o         \
224                 printf.o        \
225                 priocntl.o      \
226                 priv.o          \
227                 priv_const.o    \
228                 proc.o          \
229                 procset.o       \
230                 processor_bind.o        \
231                 processor_info.o        \
232                 profil.o        \
233                 project.o       \
234                 qsort.o         \
235                 rctl.o          \
236                 rctlsys.o       \
237                 readlink.o      \
238                 refstr.o        \
239                 rename.o        \
240                 resolvepath.o   \
241                 process.o       \
242                 rlimit.o        \
243                 rmap.o          \
244                 rmdir.o         \
245                 rw.o            \
246                 rwstlock.o      \
247                 sad_conf.o      \
248                 sched.o         \
249                 schedctl.o      \
250                 seg_dev.o       \
251                 seg_kp.o        \
252                 seg_kpm.o       \
253                 seg_map.o       \
254                 seg_vn.o        \
255                 seg_spt.o       \
256                 semaphore.o     \
257                 sendfile.o      \
```

```
258                 session.o       \
259                 share.o         \
260                 shuttle.o       \
261                 sig.o           \
262                 sigaction.o     \
263                 sigaltstack.o   \
264                 signotify.o     \
265                 sigpending.o    \
266                 sigprocmask.o   \
267                 sigqueue.o      \
268                 sigsendset.o    \
269                 sigsuspend.o    \
270                 sigtimedwait.o  \
271                 sleepq.o        \
272                 softint.o       \
273                 space.o         \
274                 sscanf.o        \
275                 ssig.o          \
276                 stat.o          \
277                 statfs.o        \
278                 statvfs.o       \
279                 str_conf.o      \
280                 strcalls.o      \
281                 stream.o        \
282                 streamio.o      \
283                 strext.o        \
284                 strsubr.o       \
285                 strsun.o        \
286                 subr.o          \
287                 sunddi.o        \
288                 sunmdi.o        \
289                 sunndi.o        \
290                 sunpci.o        \
291                 sunpm.o         \
292                 sundlpi.o       \
293                 suntpi.o        \
294                 swap_subr.o     \
295                 swap_vnops.o    \
296                 symlink.o       \
297                 sync.o          \
298                 sysclass.o      \
299                 sysconfig.o     \
300                 sysent.o        \
301                 sysfs.o         \
302                 systeminfo.o    \
303                 task.o          \
304                 taskq.o         \
305                 tasksys.o       \
306                 time.o          \
307                 timer.o         \
308                 times.o         \
309                 timers.o        \
310                 thread.o        \
311                 tlabel.o        \
312                 tnf_res.o       \
313                 turnstile.o     \
314                 tty_common.o    \
315                 uadmin.o        \
316                 uconv.o         \
317                 ucredsys.o      \
318                 uid.o           \
319                 umask.o         \
320                 umount.o        \
321                 uname.o         \
322                 unix_bb.o       \
323                 unlink.o        \
```



```
324                 urw.o           \
325                 utime.o         \
326                 utssys.o        \
327                 uucopy.o        \
328                 vfs.o           \
329                 vfs_conf.o      \
330                 vmem.o          \
331                 vm_anon.o       \
332                 vm_as.o         \
333                 vm_meter.o      \
334                 vm_pageout.o    \
335                 vm_pvn.o        \
336                 vm_rm.o         \
337                 vm_seg.o        \
338                 vm_subr.o       \
339                 vm_swap.o       \
340                 vm_usage.o      \
341                 vnode.o         \
342                 vuid_queue.o    \
343                 vuid_store.o    \
344                 watchpoint.o    \
345                 yield.o         \
346                 scsi_confdata.o \
347                 xdr_mblk.o      \
348                 xdr_mem.o       \
349                 xdr.o           \
350                 xdr_array.o     \
351                 xdr_refer.o     \
352                 xhat.o          \
353                 zone.o

355 #
356 #       Stubs for the stand-alone linker/loader
357 #
358 sparc_GENSTUBS_OBJS =   \
359         kobj_stubs.o

361 i386_GENSTUBS_OBJS =

363 COMMON_GENSTUBS_OBJS =

365 GENSTUBS_OBJS += $(COMMON_GENSTUBS_OBJS) $($(MACH)_GENSTUBS_OBJS)

367 #
368 #       DTrace and DTrace Providers
369 #
370 DTRACE_OBJS += dtrace.o dtrace_isa.o dtrace_asm.o

372 SDT_OBJS += sdt_subr.o

374 PROFILE_OBJS += profile.o

376 SYSTRACE_OBJS += systrace.o

378 LX_SYSTRACE_OBJS += lx_systrace.o

380 LOCKSTAT_OBJS += lockstat.o

382 FASTTRAP_OBJS += fasttrap.o fasttrap_isa.o

384 #
385 #       Driver (pseudo-driver) Modules
386 #
387 IPP_OBJS +=     ippctl.o

389 AUDIOVIA823X_OBJS += audiovia823x.o
```

```
391 AUDIO810_OBJS += audio810.o

393 AUDIOHD_OBJS += audiohd.o

395 AUDIOIXP_OBJS += audioixp.o

397 AUDIOTS_OBJS += audiots.o

399 CARDBUS_OBJS += cardbus.o cardbus_hp.o cardbus_cfg.o

401 CONSKBD_OBJS += conskbd.o

403 CONSMS_OBJS +=  consms.o

405 DIAUDIO_OBJS += audio.o

407 OLDPTY_OBJS +=  tty_ptyconf.o

409 PTC_OBJS +=     tty_pty.o

411 PTSL_OBJS +=    tty_pts.o

413 PTM_OBJS +=     ptm.o

415 LX_PTM_OBJS +=  lx_ptm.o

417 LX_AUDIO_OBJS += lx_audio.o

419 PTS_OBJS +=     pts.o

421 PTY_OBJS +=     ptms_conf.o

423 SAD_OBJS +=     sad.o

425 MD5_OBJS +=     md5.o md5_mod.o

427 SHA1_OBJS +=    sha1.o sha1_mod.o

429 SHA2_OBJS +=    sha2.o sha2_mod.o

431 IPGPC_OBJS +=   classifierddi.o classifier.o filters.o trie.o table.o \
432                 ba_table.o

434 DSCPMK_OBJS +=  dscpmk.o dscpmkddi.o

436 DLCOSMK_OBJS += dlcosmk.o dlcosmkddi.o

438 FLOWACCT_OBJS +=        flowacctddi.o flowacct.o

440 TOKENMT_OBJS += tokenmt.o tokenmtddi.o

442 TSWTCL_OBJS +=  tswtcl.o tswtclddi.o

444 ARP_OBJS +=     arpddi.o arp.o arp_netinfo.o

446 ICMP_OBJS +=    icmpddi.o icmp.o icmp_opt_data.o

448 ICMP6_OBJS +=   icmp6ddi.o

450 RTS_OBJS +=     rtsddi.o rts.o rts_opt_data.o

452 IP_TCP_OBJS =   tcp.o tcp_fusion.o tcp_kssl.o tcp_opt_data.o tcp_sack.o \
453                 tcp_trace.o
454 IP_UDP_OBJS =   udp.o udp_opt_data.o
455 IP_SCTP_OBJS =  sctp_crc32.o sctp.o sctp_opt_data.o sctp_output.o \
```



```
456                 sctp_init.o sctp_input.o sctp_cookie.o \
457                 sctp_conn.o sctp_error.o sctp_snmp.o \
458                 sctp_param.o sctp_shutdown.o sctp_common.o \
459                 sctp_timer.o sctp_heartbeat.o sctp_hash.o \
460                 sctp_ioc.o sctp_bind.o sctp_notify.o sctp_asconf.o \
461                 sctp_addr.o tn_ipopt.o tnet.o ip_netinfo.o

463 IP_OBJS +=      igmp.o ip.o ip6.o ip6_asp.o ip6_if.o ip6_ire.o ip6_rts.o \
464                 ip_cksum.o ip_if.o ip_ire.o ip_listutils.o ip_mroute.o \
465                 ip_multi.o ip_ndp.o ip_opt_data.o ip_rts.o ip_srcid.o \
466                 ipddi.o ipdrop.o mi.o nd.o optcom.o snmpcom.o ipsec_loader.o \
467                 spd.o ipclassifier.o inet_common.o ip_squeue.o squeue.o \
468                 ip_sadb.o ip_ftable.o radix.o \
469                 $(IP_TCP_OBJS) \
470                 $(IP_UDP_OBJS) \
471                 $(IP_SCTP_OBJS)

473 IP6_OBJS +=     ip6ddi.o

475 HOOK_OBJS +=    hook.o

477 NETI_OBJS +=    neti.o

479 KEYSOCK_OBJS += keysockddi.o keysock.o keysock_opt_data.o

481 SPDSOCK_OBJS += spdsockddi.o spdsock.o spdsock_opt_data.o

483 IPSECESP_OBJS += ipsecespddi.o ipsecesp.o

485 IPSECAH_OBJS += ipsecahddi.o ipsecah.o sadb.o

487 NATTYMOD_OBJS += nattymod.o

489 SPPP_OBJS +=    sppp.o sppp_dlpi.o sppp_mod.o s_common.o

491 SPPPTUN_OBJS += sppptun.o sppptun_mod.o

493 SPPPASYN_OBJS += spppasyn.o spppasyn_mod.o

495 SPPPCOMP_OBJS += spppcomp.o spppcomp_mod.o deflate.o bsd-comp.o vjcompress.o \
496                 zlib.o

498 TCP_OBJS +=     tcpddi.o

500 TCP6_OBJS +=    tcp6ddi.o

502 SCTP_OBJS +=    sctpddi.o

504 SCTP6_OBJS +=   sctp6ddi.o

506 NCA_OBJS +=     ncaddi.o

508 TUN_OBJS +=     tun.o

510 ATUN_OBJS +=    atun.o

512 6TO4TUN_OBJS += 6to4tun.o

514 RDS_OBJS +=     rdsddi.o rdssubr.o rds_opt.o rds_ioctl.o

516 RDSIB_OBJS +=   rdsib.o rdsib_ib.o rdsib_cm.o rdsib_ep.o rdsib_buf.o \
517                 rdsib_arp.o rdsib_arp_link.o rdsib_debug.o rdsib_sc.o

519 UDP_OBJS +=     udpddi.o

521 UDP6_OBJS +=    udp6ddi.o
```

```
523 SY_OBJS +=      gentty.o
525 TCO_OBJS +=     ticots.o
527 TCOO_OBJS +=    ticotsord.o
529 TCL_OBJS +=     ticlts.o
531 TL_OBJS +=      tl.o
533 DUMP_OBJS +=    dump.o
535 CLONE_OBJS +=   clone.o
537 CN_OBJS +=      cons.o
539 DLD_OBJS +=     dld_drv.o dld_proto.o dld_str.o
541 DLS_OBJS +=     dls.o dls_link.o dls_mod.o dls_stat.o dls_vlan.o dls_soft_ring.o
543 GLD_OBJS +=     gld.o gldutil.o
545 MAC_OBJS +=     mac.o mac_mod.o mac_stat.o
547 MAC_ETHER_OBJS +=       mac_ether.o
549 MAC_WIFI_OBJS +=        mac_wifi.o
551 AGGR_OBJS +=    aggr_dev.o aggr_ctl.o aggr_grp.o aggr_port.o \
552                 aggr_send.o aggr_recv.o aggr_lacp.o
554 NET80211_OBJS += net80211.o net80211_proto.o net80211_input.o \
555                  net80211_output.o net80211_node.o net80211_crypto.o \
556                  net80211_crypto_none.o net80211_crypto_wep.o net80211_ioctl.o
558 IB_OBJS +=      ibnex.o ibnex_ioctl.o
560 IBCM_OBJS +=    ibcm_impl.o ibcm_sm.o ibcm_ti.o ibcm_utils.o ibcm_path.o
562 IBDM_OBJS +=    ibdm.o
564 IBMF_OBJS +=    ibmf.o ibmf_impl.o ibmf_dr.o ibmf_wqe.o ibmf_ud_dest.o ibmf_mod.
565                 ibmf_send.o ibmf_recv.o ibmf_handlers.o ibmf_trans.o \
566                 ibmf_timers.o ibmf_msg.o ibmf_utils.o ibmf_rmpp.o \
567                 ibmf_saa.o ibmf_saa_impl.o ibmf_saa_utils.o ibmf_saa_events.o
569 IBTL_OBJS +=    ibtl_impl.o ibtl_util.o ibtl_mem.o ibtl_handlers.o ibtl_qp.o \
570                 ibtl_cq.o ibtl_wr.o ibtl_hca.o ibtl_chan.o ibtl_cm.o \
571                 ibtl_mcg.o ibtl_ibnex.o ibtl_srq.o
573 KSTAT_OBJS +=   kstat.o
575 KSYMS_OBJS +=   ksyms.o
577 INSTANCE_OBJS += inst_sync.o
579 IWSCN_OBJS +=   iwscons.o
581 LOFI_OBJS +=    lofi.o
583 FSSNAP_OBJS +=  fssnap.o
585 FSSNAPIF_OBJS += fssnap_if.o
587 MM_OBJS +=      mem.o
```



```
589 PHYSMEM_OBJS += physmem.o
591 OPTIONS_OBJS += options.o
593 WINLOCK_OBJS += winlockio.o
595 PM_OBJS +=      pm.o
597 PSEUDO_OBJS +=  pseudonex.o
599 RAMDISK_OBJS += ramdisk.o
601 LLC1_OBJS += llc1.o
603 USBKBM_OBJS += usbkbm.o
605 BOFI_OBJS += bofi.o
607 HID_OBJS += hid.o
609 USBSKEL_OBJS += usbskel.o
611 USBVC_OBJS += usbvc.o usbvc_v4l2.o
613 HIDPARSER_OBJS += hidparser.o
615 USB_AC_OBJS += usb_ac.o
617 USB_AC_DACF_OBJS += usb_ac_dacf.o
619 USB_AS_OBJS += usb_as.o
621 USB_AH_OBJS += usb_ah.o
623 USBMS_OBJS += usbms.o
625 USBPRN_OBJS += usbprn.o
627 UGEN_OBJS += ugen.o
629 USBSER_OBJS += usbser.o usbser_rseq.o
631 USBSACM_OBJS += usbsacm.o
633 USBSER_KEYSPAN_OBJS += usbser_keyspan.o keyspan_dsd.o keyspan_pipe.o
635 USBS49_FW_OBJS += keyspan_49fw.o
637 USBSPRL_OBJS += usbser_pl2303.o pl2303_dsd.o
639 WC_OBJS += wscons.o
641 SCSI_OBJS +=    scsi_capabilities.o     scsi_control.o  scsi_watch.o \
642                 scsi_data.o     scsi_resource.o         scsi_subr.o     \
643                 scsi_hba.o      scsi_transport.o        scsi_confsubr.o \
644                 scsi_reset_notify.o
646 SGEN_OBJS +=    sgen.o
648 SATA_OBJS +=    sata.o
650 USBA_OBJS +=    hcdi.o  usba.o  usbai.o hubdi.o parser.o genconsole.o \
651                 usbai_pipe_mgmt.o usbai_req.o usbai_util.o usbai_register.o \
652                 usba_devdb.o usba10_calls.o usba_ugen.o
```

```
654 USBA10_OBJS +=  usba10.o

656 RSM_OBJS +=     rsm.o   rsmka_pathmanager.o     rsmka_util.o

658 RSMOPS_OBJS +=  rsmops.o

660 S1394_OBJS +=   t1394.o t1394_errmsg.o s1394.o s1394_addr.o s1394_asynch.o \
661                 s1394_bus_reset.o s1394_cmp.o s1394_csr.o s1394_dev_disc.o \
662                 s1394_fa.o s1394_fcp.o \
663                 s1394_hotplug.o s1394_isoch.o s1394_misc.o h1394.o nx1394.o

665 HCI1394_OBJS += hci1394.o hci1394_async.o hci1394_attach.o hci1394_buf.o \
666                 hci1394_csr.o hci1394_detach.o hci1394_extern.o \
667                 hci1394_ioctl.o hci1394_isoch.o hci1394_isr.o \
668                 hci1394_ixl_comp.o hci1394_ixl_isr.o hci1394_ixl_misc.o \
669                 hci1394_ixl_update.o hci1394_misc.o hci1394_ohci.o \
670                 hci1394_q.o hci1394_s1394if.o hci1394_tlabel.o \
671                 hci1394_tlist.o hci1394_vendor.o

673 AV1394_OBJS +=  av1394.o av1394_as.o av1394_async.o av1394_cfgrom.o \
674                 av1394_cmp.o av1394_fcp.o av1394_isoch.o av1394_isoch_chan.o \
675                 av1394_isoch_recv.o av1394_isoch_xmit.o av1394_list.o \
676                 av1394_queue.o

678 DCAM1394_OBJS += dcam.o dcam_frame.o dcam_param.o dcam_reg.o \
679                 dcam_ring_buff.o

681 SCSA1394_OBJS += hba.o sbp2_driver.o sbp2_bus.o

683 SBP2_OBJS += cfgrom.o sbp2.o

685 PMODEM_OBJS += pmodem.o pmodem_cis.o cis.o cis_callout.o cis_handlers.o cis_para

687 ST_OBJS +=      st.o    st_conf.o

689 SYSMSG_OBJS +=  sysmsg.o

691 SES_OBJS +=     ses.o ses_sen.o ses_safte.o ses_ses.o

693 TNF_OBJS +=     tnf_buf.o       tnf_trace.o     tnf_writer.o    trace_init.o \
694                 trace_funcs.o   tnf_probe.o     tnf.o

696 LOGINDMUX_OBJS += logindmux.o

698 DEVINFO_OBJS += devinfo.o

700 DEVPOLL_OBJS += devpoll.o

702 DEVPOOL_OBJS += devpool.o

704 I8042_OBJS +=   i8042.o

706 KB8042_OBJS +=  \
707                 at_keyprocess.o \
708                 kb8042.o        \
709                 kb8042_keytables.o

711 MOUSE8042_OBJS += mouse8042.o

713 FDC_OBJS +=     fdc.o

715 ASY_OBJS +=     asy.o

717 ECPP_OBJS +=    ecpp.o

719 VUIDM3P_OBJS += vuidmice.o vuidm3p.o
```



```
721 VUIDM4P_OBJS += vuidmice.o vuidm4p.o

723 VUIDM5P_OBJS += vuidmice.o vuidm5p.o

725 VUIDPS2_OBJS += vuidmice.o vuidps2.o

727 SYSINIT_OBJS += sysinit.o sysinit_ddi.o

729 HPCSVC_OBJS += hpcsvc.o

731 PCIHPNEXUS_OBJS += pcihp.o

733 PCIEHPCNEXUS_OBJS += pciehpc.o

735 PCISHPC_OBJS += pcishpc.o

737 OPENEEPR_OBJS += openprom.o

739 RANDOM_OBJS += random.o

741 PSHOT_OBJS += pshot.o

743 GEN_DRV_OBJS += gen_drv.o

745 TCLIENT_OBJS += tclient.o

747 TPHCI_OBJS += tphci.o

749 TVHCI_OBJS += tvhci.o

751 EMUL64_OBJS += emul64.o emul64_bsd.o

753 ZCONS_OBJS += zcons.o

755 SI3124_OBJS += si3124.o

757 NSCONFIG_DEVNAME_OBJS += sdev_nsconfig_mod.o

759 AHCI_OBJS += ahci.o

761 PCIIDE_OBJS += pci-ide.o

763 PCEPP_OBJS += pcepp.o

765 CPC_OBJS += cpc.o

767 CPUID_OBJS += cpuid_drv.o

769 SYSEVENT_OBJS += sysevent.o

771 BL_OBJS += bl.o

773 DRM_OBJS += drm_sunmod.o drm_kstat.o drm_agpsupport.o drm_asm.o \
774             drm_auth.o drm_bufs.o drm_context.o drm_dma.o \
775             drm_drawable.o drm_drv.o drm_fops.o drm_ioctl.o drm_irq.o \
776             drm_lock.o drm_memory.o drm_msg.o drm_pci.o drm_scatter.o

778 #
779 #                       exec modules
780 #
781 AOUTEXEC_OBJS +=aout.o

783 ELFEXEC_OBJS += elf.o elf_notes.o old_notes.o

785 INTPEXEC_OBJS +=intp.o
```

```
787 JAVAEXEC_OBJS +=java.o

789 #
790 #                       file system modules
791 #
792 AUTOFS_OBJS +=  auto_vfsops.o auto_vnops.o auto_subr.o auto_xdr.o auto_sys.o

794 CACHEFS_OBJS += cachefs_cnode.o         cachefs_cod.o \
795                 cachefs_dir.o           cachefs_dlog.o  cachefs_filegrp.o \
796                 cachefs_fscache.o       cachefs_ioctl.o cachefs_log.o \
797                 cachefs_module.o \
798                 cachefs_noopc.o         cachefs_resource.o \
799                 cachefs_strict.o \
800                 cachefs_subr.o          cachefs_vfsops.o \
801                 cachefs_vnops.o

803 DEVFS_OBJS +=   devfs_subr.o    devfs_vfsops.o  devfs_vnops.o

805 DEV_OBJS  +=    sdev_subr.o     sdev_vfsops.o   sdev_vnops.o    \
806                 sdev_ptsops.o   sdev_comm.o     sdev_profile.o  sdev_ncache.o

808 CTFS_OBJS +=    ctfs_all.o ctfs_cdir.o ctfs_ctl.o ctfs_event.o \
809                 ctfs_latest.o ctfs_root.o ctfs_sym.o ctfs_tdir.o ctfs_tmpl.o

811 OBJFS_OBJS +=   objfs_vfs.o     objfs_root.o    objfs_common.o \
812                 objfs_odir.o    objfs_data.o

814 FDFS_OBJS +=    fdops.o

816 FIFO_OBJS +=    fifosubr.o      fifovnops.o

818 PIPE_OBJS +=    pipe.o

820 HSFS_OBJS +=    hsfs_node.o     hsfs_subr.o     hsfs_vfsops.o   hsfs_vnops.o \
821                 hsfs_susp.o     hsfs_rrip.o     hsfs_susp_subr.o

823 LOFS_OBJS +=    lofs_subr.o     lofs_vfsops.o   lofs_vnops.o

825 NAMEFS_OBJS +=  namevfs.o       namevno.o

827 NFS_OBJS +=     nfs_client.o    nfs_common.o    nfs_dump.o \
828                 nfs_subr.o      nfs_vfsops.o    nfs_vnops.o \
829                 nfs_xdr.o       nfs_sys.o       nfs_strerror.o \
830                 nfs3_vfsops.o   nfs3_vnops.o    nfs3_xdr.o \
831                 nfs_acl_vnops.o nfs_acl_xdr.o   nfs4_vfsops.o \
832                 nfs4_vnops.o    nfs4_xdr.o      nfs4_idmap.o \
833                 nfs4_shadow.o   nfs4_subr.o \
834                 nfs4_attr.o     nfs4_rnode.o    nfs4_client.o \
835                 nfs4_acache.o   nfs4_common.o   nfs4_client_state.o \
836                 nfs4_callback.o nfs4_recovery.o nfs4_client_secinfo.o \
837                 nfs4_client_debug.o     nfs_stats.o \
838                 nfs4_acl.o

840 NFSSRV_OBJS +=  nfs_server.o    nfs_srv.o       nfs3_srv.o \
841                 nfs_acl_srv.o   nfs_auth.o      nfs_auth_xdr.o \
842                 nfs_export.o    nfs_log.o       nfs_log_xdr.o \
843                 nfs4_srv.o      nfs4_state.o    nfs4_srv_attr.o \
844                 nfs4_srv_ns.o   nfs4_db.o       nfs4_srv_deleg.o \
845                 nfs4_deleg_ops.o nfs4_srv_readdir.o nfs4_dispatch.o

847 PCFS_OBJS +=    pc_alloc.o      pc_dir.o        pc_node.o       pc_subr.o \
848                 pc_vfsops.o     pc_vnops.o

850 PROC_OBJS +=    prcontrol.o     prioctl.o       prsubr.o        prusrio.o \
851                 prvfsops.o      prvnops.o
```



```
853 MNTFS_OBJS +=   mntvfsops.o     mntvnops.o

855 SPEC_OBJS +=    specsubr.o      specvfsops.o    specvnops.o

857 SOCK_OBJS +=    socksubr.o      sockvfsops.o    sockvnops.o     \
858                 socksyscalls.o  socktpi.o       sockstr.o       \
859                 socksctp.o      socksctpsubr.o  socksctpvnops.o sockssl.o \
860                 socksdp.o       socksdpsubr.o   socksdpvnops.o  \
861                 nl7c.o          nl7curi.o       nl7chttp.o      nl7clogd.o \
862                 nl7cnca.o

864 TMPFS_OBJS +=   tmp_dir.o       tmp_subr.o      tmp_tnode.o     tmp_vfsops.o \
865                 tmp_vnops.o

867 UDFS_OBJS +=    udf_alloc.o     udf_bmap.o      udf_dir.o       \
868                 udf_inode.o     udf_subr.o      udf_vfsops.o    \
869                 udf_vnops.o

871 UFS_OBJS +=     ufs_alloc.o     ufs_bmap.o      ufs_dir.o       ufs_xattr.o \
872                 ufs_inode.o     ufs_subr.o      ufs_tables.o    ufs_vfsops.o \
873                 ufs_vnops.o     quota.o         quotacalls.o    quota_ufs.o \
874                 ufs_filio.o     ufs_lockfs.o    ufs_thread.o    ufs_trans.o \
875                 ufs_acl.o       ufs_panic.o     ufs_directio.o  ufs_log.o \
876                 ufs_extvnops.o  ufs_snap.o      lufs.o          lufs_thread.o \
877                 lufs_log.o      lufs_map.o      lufs_top.o      lufs_debug.o

879 #
880 #                       LVM modules
881 #
882 MD_OBJS += md.o md_error.o md_ioctl.o md_mddb.o md_names.o \
883         md_med.o md_rename.o md_subr.o

885 MD_COMMON_OBJS = md_convert.o md_crc.o md_revchk.o

887 MD_DERIVED_OBJS = metamed_xdr.o meta_basic_xdr.o

889 SOFTPART_OBJS += sp.o sp_ioctl.o

891 STRIPE_OBJS += stripe.o stripe_ioctl.o

893 HOTSPARES_OBJS += hotspares.o

895 RAID_OBJS += raid.o raid_ioctl.o raid_replay.o raid_resync.o raid_hotspare.o

897 MIRROR_OBJS += mirror.o mirror_ioctl.o mirror_resync.o

899 NOTIFY_OBJS += md_notify.o

901 TRANS_OBJS += mdtrans.o trans_ioctl.o trans_log.o

903 ZFS_COMMON_OBJS +=              \
904         arc.o                   \
905         bplist.o                \
906         dbuf.o                  \
907         dmu.o                   \
908         dmu_send.o              \
909         dmu_object.o            \
910         dmu_objset.o            \
911         dmu_traverse.o          \
912         dmu_tx.o                \
913         dnode.o                 \
914         dnode_sync.o            \
915         dsl_dir.o               \
916         dsl_dataset.o           \
917         dsl_pool.o              \
```

```
 918         dsl_synctask.o          \
 919         dmu_zfetch.o            \
 920         dsl_prop.o              \
 921         fletcher.o              \
 922         gzip.o                  \
 923         lzjb.o                  \
 924         metaslab.o              \
 925         refcount.o              \
 926         sha256.o                \
 927         spa.o                   \
 928         spa_config.o            \
 929         spa_errlog.o            \
 930         spa_history.o           \
 931         spa_misc.o              \
 932         space_map.o             \
 933         txg.o                   \
 934         uberblock.o             \
 935         unique.o                \
 936         vdev.o                  \
 937         vdev_cache.o            \
 938         vdev_file.o             \
 939         vdev_label.o            \
 940         vdev_mirror.o           \
 941         vdev_missing.o          \
 942         vdev_queue.o            \
 943         vdev_raidz.o            \
 944         vdev_root.o             \
 945         zap.o                   \
 946         zap_leaf.o              \
 947         zap_micro.o             \
 948         zfs_byteswap.o          \
 949         zfs_fm.o                \
 950         zfs_znode.o             \
 951         zil.o                   \
 952         zio.o                   \
 953         zio_checksum.o          \
 954         zio_compress.o          \
 955         zio_inject.o

 957 ZFS_SHARED_OBJS +=              \
 958         zfs_namecheck.o         \
 959         zfs_prop.o

 961 ZFS_OBJS +=                     \
 962         $(ZFS_COMMON_OBJS)      \
 963         $(ZFS_SHARED_OBJS)      \
 964         vdev_disk.o             \
 965         zfs_acl.o               \
 966         zfs_ctldir.o            \
 967         zfs_dir.o               \
 968         zfs_ioctl.o             \
 969         zfs_log.o               \
 970         zfs_replay.o            \
 971         zfs_rlock.o             \
 972         zfs_vfsops.o            \
 973         zfs_vnops.o             \
 974         zvol.o

 976 #
 977 #                       streams modules
 978 #
 979 BUFMOD_OBJS     +=      bufmod.o

 981 CONNLD_OBJS +=  connld.o

 983 DEDUMP_OBJS +=  dedump.o
```



```
 985 DRCOMPAT_OBJS +=        drcompat.o

 987 LDLINUX_OBJS += ldlinux.o

 989 LDTERM_OBJS +=  ldterm.o uwidth.o

 991 PCKT_OBJS +=    pckt.o

 993 PFMOD_OBJS +=   pfmod.o

 995 PTEM_OBJS +=    ptem.o

 997 REDIRMOD_OBJS += strredirm.o

 999 TIMOD_OBJS +=   timod.o

1001 TIRDWR_OBJS +=  tirdwr.o

1003 TTCOMPAT_OBJS +=ttcompat.o

1005 LOG_OBJS +=     log.o

1007 PIPEMOD_OBJS += pipemod.o

1009 RPCMOD_OBJS +=  rpcmod.o        clnt_cots.o     clnt_clts.o \
1010                 clnt_gen.o      clnt_perr.o     mt_rpcinit.o    rpc_calmsg.o \
1011                 rpc_prot.o      rpc_sztypes.o   rpc_subr.o      rpcb_prot.o \
1012                 svc.o           svc_clts.o      svc_gen.o       svc_cots.o \
1013                 rpcsys.o        xdr_sizeof.o    clnt_rdma.o     svc_rdma.o \
1014                 xdr_rdma.o      rdma_subr.o     xdrrdma_sizeof.o

1016 TLIMOD_OBJS +=  tlimod.o        t_kalloc.o      t_kbind.o       t_kclose.o \
1017                 t_kconnect.o    t_kfree.o       t_kgtstate.o    t_kopen.o \
1018                 t_krcvudat.o    t_ksndudat.o    t_kspoll.o      t_kunbind.o \
1019                 t_kutil.o

1021 RLMOD_OBJS += rlmod.o

1023 TELMOD_OBJS += telmod.o

1025 CRYPTMOD_OBJS += cryptmod.o

1027 KB_OBJS +=      kbd.o           keytables.o

1029 #
1030 #                       scheduling class modules
1031 #
1032 RT_OBJS +=              rt.o
1033 RT_DPTBL_OBJS +=        rt_dptbl.o

1035 TS_OBJS +=              ts.o
1036 TS_DPTBL_OBJS +=        ts_dptbl.o

1038 IA_OBJS +=              ia.o

1040 FSS_OBJS +=             fss.o

1042 FX_OBJS +=              fx.o
1043 FX_DPTBL_OBJS +=        fx_dptbl.o

1045 #
1046 #                       Inter-Process Communication (IPC) modules
1047 #
1048 IPC_OBJS +=     ipc.o
```

```
1050 IPCMSG_OBJS +=  msg.o

1052 IPCSEM_OBJS +=  sem.o

1054 IPCSHM_OBJS +=  shm.o

1056 #
1057 #                       kernel cryptographic framework
1058 #
1059 KCF_OBJS +=     kcf.o kcf_callprov.o kcf_cbufcall.o kcf_cipher.o kcf_crypto.o \
1060                 kcf_cryptoadm.o kcf_ctxops.o kcf_digest.o kcf_dual.o \
1061                 kcf_keys.o kcf_mac.o kcf_mech_tabs.o kcf_miscapi.o \
1062                 kcf_object.o kcf_policy.o kcf_prov_tabs.o kcf_sched.o \
1063                 kcf_session.o kcf_sign.o kcf_spi.o kcf_verify.o kcf_random.o

1065 CRYPTOADM_OBJS += cryptoadm.o

1067 CRYPTO_OBJS +=  crypto.o

1069 DPROV_OBJS +=   dprov.o

1071 DCA_OBJS +=     dca.o dca_3des.o dca_debug.o dca_dsa.o dca_kstat.o dca_rng.o \
1072                 dca_rsa.o

1074 AESPROV_OBJS += aes.o aes_cbc_crypt.o aes_impl.o

1076 ARCFOURPROV_OBJS += arcfour.o arcfour_crypt.o

1078 BLOWFISHPROV_OBJS += blowfish.o blowfish_impl.o blowfish_cbc_crypt.o

1080 COMMON_RSAPROV_OBJS += rsa.o bignumimpl.o rsa_impl.o

1082 RSAPROV_OBJS += $(COMMON_RSAPROV_OBJS) $(RSAPROV_PSR_OBJS)

1084 SWRANDPROV_OBJS += swrand.o

1086 #
1087 #                       kernel SSL
1088 #
1089 KSSL_OBJS +=    kssl.o kssl_ioctl.o ksslapi.o ksslrec.o

1091 #
1092 #                       misc. modules
1093 #

1095 AMSRC2_OBJS +=  am_src2.o

1097 AUDIO_SUP_OBJS +=       audio_support.o

1099 MIXER_OBJS +=   am_main.o am_ad.o am_ioctl.o

1101 C2AUDIT_OBJS += adr.o audit.o audit_event.o audit_io.o \
1102                 audit_path.o audit_start.o audit_syscalls.o audit_token.o \
1103                 audit_mem.o audit_zone.o

1105 PCIC_OBJS +=    pcic.o

1107 PEM_OBJS +=     pem.o

1109 RPCSEC_OBJS +=  secmod.o        sec_clnt.o      sec_svc.o       sec_gen.o \
1110                 auth_des.o      auth_kern.o     auth_loopb.o\
1111                 authdesprt.o    authdesubr.o    authu_prot.o \
1112                 key_call.o      key_prot.o      svc_authu.o     svcauthdes.o

1114 RPCSEC_GSS_OBJS +=      rpcsec_gssmod.o rpcsec_gss.o rpcsec_gss_misc.o \
1115                 rpcsec_gss_utils.o svc_rpcsec_gss.o
```



```
1117 CONSCONFIG_OBJS += consconfig.o

1119 CONSCONFIG_DACF_OBJS  += consconfig_dacf.o consplat.o

1121 TEM_OBJS += tem.o tem_safe.o 6x10.o 7x14.o 12x22.o

1123 KBTRANS_OBJS +=                          \
1124                 kbtrans.o               \
1125                 kbtrans_keytables.o     \
1126                 kbtrans_polled.o        \
1127                 kbtrans_streams.o       \
1128                 usb_keytables.o

1130 KGSSD_OBJS +=   gssd_clnt_stubs.o gssd_handle.o gssd_prot.o \
1131                 gss_display_name.o gss_release_name.o gss_import_name.o \
1132                 gss_release_buffer.o gss_release_oid_set.o gen_oids.o gssdmod.o

1134 KGSSD_DERIVED_OBJS = gssd_xdr.o

1136 KGSS_DUMMY_OBJS += dmech.o

1138 CRYPTO= cksumtypes.o decrypt.o encrypt.o encrypt_length.o etypes.o \
1139         nfold.o verify_checksum.o prng.o block_size.o make_checksum.o\
1140         checksum_length.o hmac.o default_state.o mandatory_sumtype.o

1142 # crypto/des
1143 CRYPTO_DES= f_cbc.o f_cksum.o f_parity.o weak_key.o d3_cbc.o ef_crypto.o

1145 CRYPTO_DK= checksum.o derive.o dk_decrypt.o dk_encrypt.o

1147 CRYPTO_ARCFOUR= k5_arcfour.o

1149 # crypto/enc_provider
1150 CRYPTO_ENC= des.o des3.o arcfour_provider.o aes_provider.o

1152 # crypto/hash_provider
1153 CRYPTO_HASH= hash_kef_generic.o hash_kmd5.o hash_crc32.o hash_ksha1.o

1155 # crypto/keyhash_provider
1156 CRYPTO_KEYHASH= descbc.o k5_kmd5des.o k_hmac_md5.o

1158 # crypto/crc32
1159 CRYPTO_CRC32= crc32.o

1161 # crypto/old
1162 CRYPTO_OLD= old_decrypt.o old_encrypt.o

1164 # crypto/raw
1165 CRYPTO_RAW= raw_decrypt.o raw_encrypt.o

1167 K5_KRB= kfree.o copy_key.o \
1168         parse.o init_ctx.o \
1169         ser_adata.o ser_addr.o \
1170         ser_auth.o ser_cksum.o \
1171         ser_key.o ser_princ.o \
1172         serialize.o unparse.o \
1173         ser_actx.o

1175 K5_OS=  timeofday.o toffset.o \
1176         init_os_ctx.o c_ustime.o

1178 SEAL=
1179 # EXPORT DELETE START
1180 SEAL=   seal.o unseal.o
1181 # EXPORT DELETE END
```

```
1183 MECH=   delete_sec_context.o \
1184         import_sec_context.o \
1185         gssapi_krb5.o \
1186         k5seal.o k5unseal.o k5sealv3.o \
1187         ser_sctx.o \
1188         sign.o \
1189         util_crypt.o  \
1190         util_validate.o  util_ordering.o  \
1191         util_seqnum.o util_set.o util_seed.o \
1192         wrap_size_limit.o verify.o



1196 MECH_GEN= util_token.o


1199 KGSS_KRB5_OBJS += krb5mech.o \
1200         $(MECH) $(SEAL) $(MECH_GEN) \
1201         $(CRYPTO) $(CRYPTO_DES) $(CRYPTO_DK) $(CRYPTO_ARCFOUR) \
1202         $(CRYPTO_ENC) $(CRYPTO_HASH) \
1203         $(CRYPTO_KEYHASH) $(CRYPTO_CRC32) \
1204         $(CRYPTO_OLD) \
1205         $(CRYPTO_RAW) $(K5_KRB) $(K5_OS)

1207 DES_OBJS +=     des_crypt.o des_cbc_crypt.o des_impl.o des_ks.o des_soft.o

1209 DLBOOT_OBJS +=  bootparam_xdr.o nfs_dlinet.o scan.o

1211 sparc_KRTLD_OBJS =      \
1212         kobj_subr.o

1214 i386_KRTLD_OBJS =

1216 COMMON_KRTLD_OBJS =     \
1217         kobj_bootflags.o \
1218         getoptstr.o     \
1219         kobj.o          \
1220         kobj_kdi.o      \
1221         kobj_lm.o

1223 KRTLD_OBJS +=   $(COMMON_KRTLD_OBJS) $($(MACH)_KRTLD_OBJS)

1225 MOD_OBJS +=     modctl.o modsubr.o modsysfile.o modconf.o modhash.o

1227 STRPLUMB_OBJS += strplumb.o octet.o

1229 CPR_OBJS +=     cpr_driver.o cpr_dump.o \
1230                 cpr_main.o cpr_misc.o cpr_mod.o cpr_stat.o \
1231                 cpr_uthread.o

1233 PROF_OBJS +=    prf.o

1235 SE_OBJS += se_driver.o

1237 SYSACCT_OBJS += acct.o

1239 ACCTCTL_OBJS += acctctl.o

1241 EXACCTSYS_OBJS += exacctsys.o

1243 KAIO_OBJS += aio.o

1245 PCMCIA_OBJS += pcmcia.o cs.o cis.o cis_callout.o cis_handlers.o cis_params.o

1247 BUSRA_OBJS += busra.o
```



```
1249 PCS_OBJS += pcs.o

1251 PCATA_OBJS += pcide.o pcdisk.o pclabel.o pcata.o

1253 PCMEM_OBJS += pcmem.o

1255 PCRAM_OBJS += pcram.o

1257 PCSER_OBJS += pcser.o pcser_cis.o

1259 PSET_OBJS +=    pset.o

1261 PCI_I2ONEXUS_OBJS += pci_to_i2o.o

1263 I2OMSG_OBJS += i2o_msg.o

1265 I2O_SCSI_OBJS += i2o_scsi.o

1267 I2O_BS_OBJS += i2o_bs.o label.o

1269 OHCI_OBJS += ohci.o ohci_hub.o ohci_polled.o

1271 UHCI_OBJS += uhci.o uhciutil.o uhcitgt.o uhcihub.o uhcipolled.o

1273 EHCI_OBJS += ehci.o ehci_hub.o ehci_xfer.o ehci_intr.o ehci_util.o ehci_polled.o

1275 HUBD_OBJS += hubd.o

1277 USB_MID_OBJS += usb_mid.o

1279 USB_IA_OBJS += usb_ia.o

1281 SCSA2USB_OBJS += scsa2usb.o usb_ms_bulkonly.o usb_ms_cbi.o

1283 PHX_OBJS += phx.o

1285 IPF_OBJS += ip_fil_solaris.o fil.o solaris.o ip_state.o ip_frag.o ip_nat.o \
1286             ip_proxy.o ip_auth.o ip_pool.o ip_htable.o ip_lookup.o \
1287             ip_log.o misc.o
1288
1289 IBD_OBJS +=     ibd.o

1291 SDP_OBJS +=     sdpddi.o

1293 VNI_OBJS += vni.o

1295 CTF_OBJS += ctf_create.o ctf_decl.o ctf_error.o ctf_hash.o ctf_labels.o \
1296         ctf_lookup.o ctf_open.o ctf_types.o ctf_util.o ctf_subr.o ctf_mod.o

1297 ZMOD_OBJS += adler32.o infblock.o infcodes.o inffast.o inflate.o inftrees.o \
1298                 infutil.o zutil.o zmod.o

1298 SMBIOS_OBJS += smb_error.o smb_info.o smb_open.o smb_subr.o smb_dev.o

1300 RPCIB_OBJS += rpcib.o

1302 KMDB_OBJS += kdrv.o

1304 BGE_OBJS += bge_main2.o bge_chip2.o bge_kstats.o bge_log.o bge_ndd.o \
1305                 bge_atomic.o bge_mii.o bge_send.o bge_recv2.o

1307 IXGB_OBJS += ixgb.o ixgb_atomic.o ixgb_chip.o ixgb_gld.o ixgb_kstats.o \
1308                 ixgb_log.o ixgb_ndd.o ixgb_rx.o ixgb_tx.o ixgb_xmii.o

1310 RGE_OBJS += rge_main.o rge_chip.o rge_ndd.o rge_kstats.o rge_log.o rge_rxtx.o
```

```
1312 ATH_OBJS += ath_aux.o ath_main.o ath_osdep.o ath_rate.o

1314 #
1315 #       Build up defines and paths.
1316 #
1317 LINT_DEFS       += -Dunix

1319 #
1320 #       This duality can be removed when the native and target compilers
1321 #       are the same (or at least recognize the same command line syntax!)
1322 #       It is a bug in the current compilation system that the assember
1323 #       can't process the -Y I, flag.
1324 #
1325 NATIVE_INC_PATH += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common
1326 AS_INC_PATH     += $(INC_PATH) -I$(UTSBASE)/common
1327 INCLUDE_PATH    += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common

1329 #
1330 PCIE_OBJS += pcie.o pcie_fault.o

1332 #       Chelsio N110 10G NIC driver module
1333 #
1334 CH_OBJS = ch.o glue.o pe.o sge.o

1336 CH_COM_OBJS =   ch_mac.o ch_subr.o cspi.o espi.o ixf1010.o mc3.o mc4.o mc5.o \
1337                 mv88e1xxx.o mv88x201x.o my3126.o pm3393.o tp.o ulp.o \
1338                 vsc7321.o vsc7326.o xpak.o

1340 #
1341 #       PCI strings file
1342 #
1343 PCI_STRING_OBJS = pci_strings.o

1345 #
1346 #       Xframe 10G NIC driver module
1347 #
1348 XGE_OBJS = xge.o xgell.o

1350 XGE_HAL_OBJS =  xgehal-channel.o xgehal-fifo.o xgehal-ring.o  xgehal-config.o \
1351                 xgehal-driver.o  xgehal-mm.o xgehal-stats.o  xgehal-device.o \
1352                 xge-queue.o  xgehal-mgmt.o xgehal-mgmtaux.o
```

```
**********************************************************
   41637 Wed Jan 31 21:21:16 2007
new/usr/src/uts/common/Makefile.rules
1000001 add zlib compression support
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#


#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)Makefile.rules     1.345   07/01/31 SMI"
# ident "@(#)Makefile.rules     1.344   07/01/21 SMI"
#
# uts/common/Makefile.rules
#
#       This Makefile defines all the file build rules for the directory
# uts/common and its children. These are the source files which may
# be considered common to all SunOS systems.
#
#       The following two-level ordering must be maintained in this file.
#         Lines are sorted first in order of decreasing specificity based on
#         the first directory component.  That is, sun4u rules come before
#         sparc rules come before common rules.
#
#         Lines whose initial directory components are equal are sorted
#         alphabetically by the remaining components.


#
#       Section 1a: C objects build rules
#
$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/aes/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/arcfour/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/blowfish/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/rsa/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)
```



```
$(OBJS_DIR)/%.o:                $(COMMONBASE)/bignum/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/acl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/avl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/brand/sn1/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/c2/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/conf/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/contract/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/cpr/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/ctf/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/ctf/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/des/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/smbios/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/des/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/crypto/api/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/crypto/core/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/crypto/io/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/crypto/spi/%.c
        $(COMPILE.c) -o $@ $<
```

```
127         $(CTFCONVERT_O)

129 $(OBJS_DIR)/%.o:                $(COMMONBASE)/pci/%.c
130         $(COMPILE.c) -o $@ $<
131         $(CTFCONVERT_O)

133 $(OBJS_DIR)/%.o:                $(COMMONBASE)/devid/%.c
134         $(COMPILE.c) -o $@ $<
135         $(CTFCONVERT_O)

137 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/disp/%.c
138         $(COMPILE.c) -o $@ $<
139         $(CTFCONVERT_O)

141 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/dtrace/%.c
142         $(COMPILE.c) -o $@ $<
143         $(CTFCONVERT_O)

145 $(OBJS_DIR)/%.o:                $(COMMONBASE)/exacct/%.c
146         $(COMPILE.c) -o $@ $<
147         $(CTFCONVERT_O)

149 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/exec/aout/%.c
150         $(COMPILE.c) -o $@ $<
151         $(CTFCONVERT_O)

153 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/exec/elf/%.c
154         $(COMPILE.c) -o $@ $<
155         $(CTFCONVERT_O)

157 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/exec/intp/%.c
158         $(COMPILE.c) -o $@ $<
159         $(CTFCONVERT_O)

161 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/exec/java/%.c
162         $(COMPILE.c) -o $@ $<
163         $(CTFCONVERT_O)

165 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/%.c
166         $(COMPILE.c) -o $@ $<
167         $(CTFCONVERT_O)

169 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/autofs/%.c
170         $(COMPILE.c) -o $@ $<
171         $(CTFCONVERT_O)

173 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/cachefs/%.c
174         $(COMPILE.c) -o $@ $<
175         $(CTFCONVERT_O)

177 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/devfs/%.c
178         $(COMPILE.c) -o $@ $<
179         $(CTFCONVERT_O)

181 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/ctfs/%.c
182         $(COMPILE.c) -o $@ $<
183         $(CTFCONVERT_O)

185 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/doorfs/%.c
186         $(COMPILE.c) -o $@ $<
187         $(CTFCONVERT_O)

189 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/dev/%.c
190         $(COMPILE.c) -o $@ $<
191         $(CTFCONVERT_O)
```



```
193 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/fd/%.c
194         $(COMPILE.c) -o $@ $<
195         $(CTFCONVERT_O)

197 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/fifofs/%.c
198         $(COMPILE.c) -o $@ $<
199         $(CTFCONVERT_O)

201 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/hsfs/%.c
202         $(COMPILE.c) -o $@ $<
203         $(CTFCONVERT_O)

205 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/lofs/%.c
206         $(COMPILE.c) -o $@ $<
207         $(CTFCONVERT_O)

209 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/mntfs/%.c
210         $(COMPILE.c) -o $@ $<
211         $(CTFCONVERT_O)

213 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/namefs/%.c
214         $(COMPILE.c) -o $@ $<
215         $(CTFCONVERT_O)

217 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/nfs/%.c
218         $(COMPILE.c) -o $@ $<
219         $(CTFCONVERT_O)

221 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/objfs/%.c
222         $(COMPILE.c) -o $@ $<
223         $(CTFCONVERT_O)

225 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/pcfs/%.c
226         $(COMPILE.c) -o $@ $<
227         $(CTFCONVERT_O)

229 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/portfs/%.c
230         $(COMPILE.c) -o $@ $<
231         $(CTFCONVERT_O)

233 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/proc/%.c
234         $(COMPILE.c) -o $@ $<
235         $(CTFCONVERT_O)

237 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/sockfs/%.c
238         $(COMPILE.c) -o $@ $<
239         $(CTFCONVERT_O)

241 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/specfs/%.c
242         $(COMPILE.c) -o $@ $<
243         $(CTFCONVERT_O)

245 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/swapfs/%.c
246         $(COMPILE.c) -o $@ $<
247         $(CTFCONVERT_O)

249 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/tmpfs/%.c
250         $(COMPILE.c) -o $@ $<
251         $(CTFCONVERT_O)

253 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/udfs/%.c
254         $(COMPILE.c) -o $@ $<
255         $(CTFCONVERT_O)

257 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/ufs/%.c
258         $(COMPILE.c) -o $@ $<
```

```
259         $(CTFCONVERT_O)

261 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/zfs/%.c
262         $(COMPILE.c) -o $@ $<
263         $(CTFCONVERT_O)

265 $(OBJS_DIR)/%.o:                $(COMMONBASE)/zfs/%.c
266         $(COMPILE.c) -o $@ $<
267         $(CTFCONVERT_O)

269 KMECHKRB5_BASE=$(UTSBASE)/common/gssapi/mechs/krb5

271 KGSSDFLAGS=-I $(UTSBASE)/common/gssapi/include

273 # Note, KRB5_DEFS can be assigned various preprocessor flags,
274 # typically -D defines on the make invocation.  The standard compiler
275 # flags will not be overwritten.
276 KGSSDFLAGS += $(KRB5_DEFS)

278 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/gssapi/%.c
279         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
280         $(CTFCONVERT_O)

282 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/gssapi/mechs/dummy/%.c
283         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
284         $(CTFCONVERT_O)

286 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/%.c
287         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
288         $(CTFCONVERT_O)

290 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/%.c
291         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
292         $(CTFCONVERT_O)

294 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/des/%.c
295         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
296         $(CTFCONVERT_O)

298 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/arcfour/%.c
299         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
300         $(CTFCONVERT_O)

302 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/dk/%.c
303         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
304         $(CTFCONVERT_O)

306 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/enc_provider/%.c
307         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
308         $(CTFCONVERT_O)

310 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/hash_provider/%.c
311         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
312         $(CTFCONVERT_O)

314 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/keyhash_provider/%.c
315         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
316         $(CTFCONVERT_O)

318 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/raw/%.c
319         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
320         $(CTFCONVERT_O)

322 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/crc32/%.c
323         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
324         $(CTFCONVERT_O)
```



```
326 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/old/%.c
327         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
328         $(CTFCONVERT_O)

330 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/krb5/krb/%.c
331         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
332         $(CTFCONVERT_O)

334 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/krb5/os/%.c
335         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
336         $(CTFCONVERT_O)

338 $(OBJS_DIR)/ser_sctx.o := CPPFLAGS += -DPROVIDE_KERNEL_IMPORT=1

340 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/mech/%.c
341         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
342         $(CTFCONVERT_O)

344 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/profile/%.c
345         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
346         $(CTFCONVERT_O)

348 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/%.c
349         $(COMPILE.c) -o $@ $<
350         $(CTFCONVERT_O)

352 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/arp/%.c
353         $(COMPILE.c) -o $@ $<
354         $(CTFCONVERT_O)

356 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/ip/%.c
357         $(COMPILE.c) -o $@ $<
358         $(CTFCONVERT_O)

360 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/kssl/%.c
361         $(COMPILE.c) -o $@ $<
362         $(CTFCONVERT_O)

364 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/sctp/%.c
365         $(COMPILE.c) -o $@ $<
366         $(CTFCONVERT_O)

368 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/tcp/%.c
369         $(COMPILE.c) -o $@ $<
370         $(CTFCONVERT_O)


373 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/ipf/%.c
374         $(COMPILE.c) -o $@ $<
375         $(CTFCONVERT_O)

377 $(OBJS_DIR)/%.o:                $(COMMONBASE)/net/patricia/%.c
378         $(COMPILE.c) -o $@ $<
379         $(CTFCONVERT_O)

381 IPFFLAG2=-I $(UTSBASE)/common/inet/ipf
382 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/pfil/%.c
383         $(COMPILE.c) $(IPFFLAG2) -o $@ $<
384         $(CTFCONVERT_O)

386 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/udp/%.c
387         $(COMPILE.c) -o $@ $<
388         $(CTFCONVERT_O)

390 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/nca/%.c
```

```
391         $(COMPILE.c) -o $@ $<
392         $(CTFCONVERT_O)

394 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/inet/vni/%.c
395         $(COMPILE.c) -o $@ $<
396         $(CTFCONVERT_O)

398 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/%.c
399         $(COMPILE.c) -o $@ $<
400         $(CTFCONVERT_O)

402 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/1394/%.c
403         $(COMPILE.c) -o $@ $<
404         $(CTFCONVERT_O)

406 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/1394/adapters/%.c
407         $(COMPILE.c) -o $@ $<
408         $(CTFCONVERT_O)

410 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/1394/targets/av1394/%.c
411         $(COMPILE.c) -o $@ $<
412         $(CTFCONVERT_O)

414 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/1394/targets/dcam1394/%.c
415         $(COMPILE.c) -o $@ $<
416         $(CTFCONVERT_O)

418 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/1394/targets/scsa1394/%.c
419         $(COMPILE.c) -o $@ $<
420         $(CTFCONVERT_O)

422 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/sbp2/%.c
423         $(COMPILE.c) -o $@ $<
424         $(CTFCONVERT_O)

426 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ath/%.c
427         $(COMPILE.c) -o $@ $<
428         $(CTFCONVERT_O)

430 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/audio/legacy/diaudio/%.c
431         $(COMPILE.c) -o $@ $<
432         $(CTFCONVERT_O)

434 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/audio/sada/amsrc2/%.c
435         $(COMPILE.c) -o $@ $<
436         $(CTFCONVERT_O)

438 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/audio/sada/drv/audio810/%.c
439         $(COMPILE.c) -o $@ $<
440         $(CTFCONVERT_O)

442 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/audio/sada/drv/audiohd/%.c
443         $(COMPILE.c) -o $@ $<
444         $(CTFCONVERT_O)

446 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/audio/sada/drv/audioixp/%.c
447         $(COMPILE.c) -o $@ $<
448         $(CTFCONVERT_O)

450 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/audio/sada/drv/audiots/%.c
451         $(COMPILE.c) -o $@ $<
452         $(CTFCONVERT_O)

454 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/audio/sada/framework/%.c
455         $(COMPILE.c) -o $@ $<
456         $(CTFCONVERT_O)
```



```
458 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/audio/sada/mixer/%.c
459         $(COMPILE.c) -o $@ $<
460         $(CTFCONVERT_O)

462 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/bge/%.c
463         $(COMPILE.c) -o $@ $<
464         $(CTFCONVERT_O)

466 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/cardbus/%.c
467         $(COMPILE.c) -o $@ $<
468         $(CTFCONVERT_O)

470 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/dld/%.c
471         $(COMPILE.c) -o $@ $<
472         $(CTFCONVERT_O)

474 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/dls/%.c
475         $(COMPILE.c) -o $@ $<
476         $(CTFCONVERT_O)

478 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/drm/%.c
479         $(COMPILE.c) -o $@ $<
480         $(CTFCONVERT_O)

482 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/hotplug/hpcsvc/%.c
483         $(COMPILE.c) -o $@ $<
484         $(CTFCONVERT_O)

486 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/hotplug/pciehpc/%.c
487         $(COMPILE.c) -o $@ $<
488         $(CTFCONVERT_O)

490 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/hotplug/pcishpc/%.c
491         $(COMPILE.c) -o $@ $<
492         $(CTFCONVERT_O)

494 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/hotplug/pcihp/%.c
495         $(COMPILE.c) -o $@ $<
496         $(CTFCONVERT_O)

498 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/i2o/%.c
499         $(COMPILE.c) -o $@ $<
500         $(CTFCONVERT_O)

502 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ib/clients/rds/%.c
503         $(COMPILE.c) -o $@ $<
504         $(CTFCONVERT_O)

506 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ib/clients/ibd/%.c
507         $(COMPILE.c) -o $@ $<
508         $(CTFCONVERT_O)

510 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ib/clients/sdp/%.c
511         $(COMPILE.c) -o $@ $<
512         $(CTFCONVERT_O)

514 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ib/mgt/ibcm/%.c
515         $(COMPILE.c) -o $@ $<
516         $(CTFCONVERT_O)

518 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ib/mgt/ibdm/%.c
519         $(COMPILE.c) -o $@ $<
520         $(CTFCONVERT_O)

522 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ib/mgt/ibmf/%.c
```

```
523         $(COMPILE.c) -o $@ $<
524         $(CTFCONVERT_O)

526 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ib/ibnex/%.c
527         $(COMPILE.c) -o $@ $<
528         $(CTFCONVERT_O)

530 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ib/ibtl/%.c
531         $(COMPILE.c) -o $@ $<
532         $(CTFCONVERT_O)

534 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/kb8042/%.c
535         $(COMPILE.c) -o $@ $<
536         $(CTFCONVERT_O)

538 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/kbtrans/%.c
539         $(COMPILE.c) -o $@ $<
540         $(CTFCONVERT_O)

542 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/aggr/%.c
543         $(COMPILE.c) -o $@ $<
544         $(CTFCONVERT_O)

546 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/lp/%.c
547         $(COMPILE.c) -o $@ $<
548         $(CTFCONVERT_O)

550 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/lvm/hotspares/%.c
551         $(COMPILE.c) -o $@ $<
552         $(CTFCONVERT_O)

554 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/lvm/md/%.c
555         $(COMPILE.c) -o $@ $<
556         $(CTFCONVERT_O)

558 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/lvm/mirror/%.c
559         $(COMPILE.c) -o $@ $<
560         $(CTFCONVERT_O)

562 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/lvm/notify/%.c
563         $(COMPILE.c) -o $@ $<
564         $(CTFCONVERT_O)

566 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/lvm/raid/%.c
567         $(COMPILE.c) -o $@ $<
568         $(CTFCONVERT_O)

570 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/lvm/softpart/%.c
571         $(COMPILE.c) -o $@ $<
572         $(CTFCONVERT_O)

574 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/lvm/stripe/%.c
575         $(COMPILE.c) -o $@ $<
576         $(CTFCONVERT_O)

578 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/lvm/trans/%.c
579         $(COMPILE.c) -o $@ $<
580         $(CTFCONVERT_O)

582 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/mac/%.c
583         $(COMPILE.c) -o $@ $<
584         $(CTFCONVERT_O)

586 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/mac/plugins/%.c
587         $(COMPILE.c) -o $@ $<
588         $(CTFCONVERT_O)
```



```
590 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/net80211/%.c
591         $(COMPILE.c) -o $@ $<
592         $(CTFCONVERT_O)

594 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/pci-ide/%.c
595         $(COMPILE.c) -o $@ $<
596         $(CTFCONVERT_O)

598 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/pcmcia/%.c
599         $(COMPILE.c) -o $@ $<
600         $(CTFCONVERT_O)

602 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ppp/sppp/%.c
603         $(COMPILE.c) -o $@ $<
604         $(CTFCONVERT_O)

606 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/pcmcia/pem/%.c
607         $(COMPILE.c) -o $@ $<
608         $(CTFCONVERT_O)

610 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ppp/spppasyn/%.c
611         $(COMPILE.c) -o $@ $<
612         $(CTFCONVERT_O)

614 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ppp/spppcomp/%.c
615         $(COMPILE.c) -o $@ $<
616         $(CTFCONVERT_O)

614 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/ppp/sppptun/%.c
615         $(COMPILE.c) -o $@ $<
616         $(CTFCONVERT_O)

618 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/rsm/%.c
619         $(COMPILE.c) -o $@ $<
620         $(CTFCONVERT_O)

622 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/rge/%.c
623         $(COMPILE.c) -o $@ $<
624         $(CTFCONVERT_O)

626 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/sata/adapters/ahci/%.c
627         $(COMPILE.c) -o $@ $<
628         $(CTFCONVERT_O)

630 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/sata/adapters/si3124/%.c
631         $(COMPILE.c) -o $@ $<
632         $(CTFCONVERT_O)

634 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/sata/impl/%.c
635         $(COMPILE.c) -o $@ $<
636         $(CTFCONVERT_O)

638 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/scsi/conf/%.c
639         $(COMPILE.c) -o $@ $<
640         $(CTFCONVERT_O)

642 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/scsi/impl/%.c
643         $(COMPILE.c) -o $@ $<
644         $(CTFCONVERT_O)

646 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/scsi/targets/%.c
647         $(COMPILE.c) -o $@ $<
648         $(CTFCONVERT_O)

650 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/io/scsi/adapters/%.c
```

```
651         $(COMPILE.c) -o $@ $<
652         $(CTFCONVERT_O)

654 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/audio/usb_ac/%.
655         $(COMPILE.c) -o $@ $<
656         $(CTFCONVERT_O)

658 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/audio/usb_as/%.
659         $(COMPILE.c) -o $@ $<
660         $(CTFCONVERT_O)

662 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/audio/usb_ah/%.
663         $(COMPILE.c) -o $@ $<
664         $(CTFCONVERT_O)

666 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbskel/%.c
667         $(COMPILE.c) -o $@ $<
668         $(CTFCONVERT_O)

670 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/video/usbvc/%.c
671         $(COMPILE.c) -o $@ $<
672         $(CTFCONVERT_O)

674 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/hid/%.c
675         $(COMPILE.c) -o $@ $<
676         $(CTFCONVERT_O)

678 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/hidparser/%.c
679         $(COMPILE.c) -o $@ $<
680         $(CTFCONVERT_O)

682 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/printer/%.c
683         $(COMPILE.c) -o $@ $<
684         $(CTFCONVERT_O)

686 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbkbm/%.c
687         $(COMPILE.c) -o $@ $<
688         $(CTFCONVERT_O)

690 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbms/%.c
691         $(COMPILE.c) -o $@ $<
692         $(CTFCONVERT_O)

694 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/ugen/%.c
695         $(COMPILE.c) -o $@ $<
696         $(CTFCONVERT_O)

698 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbser/%.c
699         $(COMPILE.c) -o $@ $<
700         $(CTFCONVERT_O)

702 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbser/usbsacm/
703         $(COMPILE.c) -o $@ $<
704         $(CTFCONVERT_O)

706 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbser/usbser_k
707         $(COMPILE.c) -o $@ $<
708         $(CTFCONVERT_O)

710 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbser/usbsprl/
711         $(COMPILE.c) -o $@ $<
712         $(CTFCONVERT_O)

714 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/hcd/openhci/%.c
715         $(COMPILE.c) -o $@ $<
716         $(CTFCONVERT_O)
```



```
718 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/hcd/ehci/%.c
719         $(COMPILE.c) -o $@ $<
720         $(CTFCONVERT_O)

722 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/hcd/uhci/%.c
723         $(COMPILE.c) -I../../common -o $@ $<
724         $(CTFCONVERT_O)

726 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/hubd/%.c
727         $(COMPILE.c) -o $@ $<
728         $(CTFCONVERT_O)

730 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/scsa2usb/%.c
731         $(COMPILE.c) -o $@ $<
732         $(CTFCONVERT_O)

734 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/usb_mid/%.c
735         $(COMPILE.c) -o $@ $<
736         $(CTFCONVERT_O)

738 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/usb_ia/%.c
739         $(COMPILE.c) -o $@ $<
740         $(CTFCONVERT_O)

742 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/usba/%.c
743         $(COMPILE.c) -o $@ $<
744         $(CTFCONVERT_O)

746 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/usba10/%.c
747         $(COMPILE.c) -o $@ $<
748         $(CTFCONVERT_O)

750 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/vuidmice/%.c
751         $(COMPILE.c) -o $@ $<
752         $(CTFCONVERT_O)

754 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/chxge/com/%.c
755         $(COMPILE.c) -o $@ $<
756         $(CTFCONVERT_O)

758 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/chxge/%.c
759         $(COMPILE.c) -o $@ $<
760         $(CTFCONVERT_O)

762 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ixgb/%.c
763         $(COMPILE.c) -o $@ $<
764         $(CTFCONVERT_O)

766 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/xge/drv/%.c
767         $(COMPILE.c) -o $@ $<
768         $(CTFCONVERT_O)

770 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/xge/hal/xgehal/%.c
771         $(COMPILE.c) -o $@ $<
772         $(CTFCONVERT_O)

774 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/ipp/%.c
775         $(COMPILE.c) -o $@ $<
776         $(CTFCONVERT_O)

778 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/ipp/ipgpc/%.c
779         $(COMPILE.c) -o $@ $<
780         $(CTFCONVERT_O)

782 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/ipp/dlcosmk/%.c
```

```
783         $(COMPILE.c) -o $@ $<
784         $(CTFCONVERT_O)

786 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/ipp/flowacct/%.c
787         $(COMPILE.c) -o $@ $<
788         $(CTFCONVERT_O)

790 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/ipp/dscpmk/%.c
791         $(COMPILE.c) -o $@ $<
792         $(CTFCONVERT_O)

794 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/ipp/meters/%.c
795         $(COMPILE.c) -o $@ $<
796         $(CTFCONVERT_O)

798 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/kmdb/%.c
799         $(COMPILE.c) -o $@ $<
800         $(CTFCONVERT_O)

802 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/ktli/%.c
803         $(COMPILE.c) -o $@ $<
804         $(CTFCONVERT_O)

806 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/krtld/%.c
807         $(COMPILE.c) -o $@ $<
808         $(CTFCONVERT_O)

810 $(OBJS_DIR)/%.o:                 $(COMMONBASE)/lvm/%.c
811         $(COMPILE.c) -o $@ $<
812         $(CTFCONVERT_O)

814 $(OBJS_DIR)/%.o:                 $(COMMONBASE)/crypto/md5/%.c
815         $(COMPILE.c) -o $@ $<
816         $(CTFCONVERT_O)

818 $(OBJS_DIR)/%.o:                 $(COMMONBASE)/net/dhcp/%.c
819         $(COMPILE.c) -o $@ $<
820         $(CTFCONVERT_O)

822 $(OBJS_DIR)/%.o:                 $(COMMONBASE)/nvpair/%.c
823         $(COMPILE.c) -o $@ $<
824         $(CTFCONVERT_O)

826 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/os/%.c
827         $(COMPILE.c) -o $@ $<
828         $(CTFCONVERT_O)

830 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/pcmcia/cis/%.c
831         $(COMPILE.c) -o $@ $<
832         $(CTFCONVERT_O)

834 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/pcmcia/cs/%.c
835         $(COMPILE.c) -o $@ $<
836         $(CTFCONVERT_O)

838 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/pcmcia/nexus/%.c
839         $(COMPILE.c) -o $@ $<
840         $(CTFCONVERT_O)

842 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/pcmcia/pcs/%.c
843         $(COMPILE.c) -o $@ $<
844         $(CTFCONVERT_O)

846 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/rpc/%.c
847         $(COMPILE.c) -o $@ $<
848         $(CTFCONVERT_O)
```



```
850 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/rpc/sec/%.c
851         $(COMPILE.c) -o $@ $<
852         $(CTFCONVERT_O)

854 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/rpc/sec_gss/%.c
855         $(COMPILE.c) -o $@ $<
856         $(CTFCONVERT_O)

858 $(OBJS_DIR)/%.o:                 $(COMMONBASE)/crypto/sha1/%.c
859         $(COMPILE.c) -o $@ $<
860         $(CTFCONVERT_O)

862 $(OBJS_DIR)/%.o:                 $(COMMONBASE)/crypto/sha2/%.c
863         $(COMPILE.c) -o $@ $<
864         $(CTFCONVERT_O)

866 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/syscall/%.c
867         $(COMPILE.c) -o $@ $<
868         $(CTFCONVERT_O)

870 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/tnf/%.c
871         $(COMPILE.c) -o $@ $<
872         $(CTFCONVERT_O)

874 $(OBJS_DIR)/%.o:                 $(COMMONBASE)/tsol/%.c
875         $(COMPILE.c) -o $@ $<
876         $(CTFCONVERT_O)

878 $(OBJS_DIR)/%.o:                 $(COMMONBASE)/util/%.c
879         $(COMPILE.c) -o $@ $<
880         $(CTFCONVERT_O)

882 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/vm/%.c
883         $(COMPILE.c) -o $@ $<
884         $(CTFCONVERT_O)

886 $(OBJS_DIR)/%.o:                 $(UTSBASE)/common/zmod/%.c
887         $(COMPILE.c) -o $@ $<
888         $(CTFCONVERT_O)

890 #
891 # SVM
892 #

894 MD_XDR_CSRC     =       $(UTSBASE)/common/io/lvm/md
895 MD_XDR_XSRC     =       $(UTSBASE)/common/sys/lvm
896 RPCGENFLAGS     +=      -C -M -D_KERNEL -DSYSV

898 $(MD_XDR_CSRC)/meta_basic_xdr.c:        $(MD_XDR_XSRC)/meta_basic.x
899         $(RPCGEN) $(RPCGENFLAGS) -c -i 100 $(MD_XDR_XSRC)/meta_basic.x | \
900         nawk '{sub(/^#include "(\.\.\/\.\.\/)/,"#include \"\.\.\/\.\.\/\.\.\/\.\
901         nawk '{sub(/meta_basic.h/, "md_basic.h"); print $$0}' >$@

903 $(MD_XDR_CSRC)/metamed_xdr.c:   $(MD_XDR_XSRC)/metamed.x
904         $(RPCGEN) $(RPCGENFLAGS) -c -i 100 $(MD_XDR_XSRC)/metamed.x |  \
905         nawk '{sub(/^#include "(\.\.\/\.\.\/)/,"#include \"\.\.\/\.\.\/\.\.\/\.\
906         nawk '{sub(/metamed.h/, "mdmed.h"); print $$0}' >$@

908 #
909 # Privilege constants
910 #

912 PRIVS_C = $(UTSBASE)/common/os/priv_const.c

914 $(PRIVS_C): $(PRIVS_AWK) $(PRIVS_DEF)
```

```
915         $(NAWK) -f $(PRIVS_AWK) < $(PRIVS_DEF) cfile=$@

917 #
918 #       Section 1b:     Lint 'objects'
919 #
920 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/aes/%.c
921         @($(LHEAD) $(LINT.c) $< $(LTAIL))

923 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/arcfour/%.c
924         @($(LHEAD) $(LINT.c) $< $(LTAIL))

926 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/blowfish/%.c
927         @($(LHEAD) $(LINT.c) $< $(LTAIL))

929 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/rsa/%.c
930         @($(LHEAD) $(LINT.c) $< $(LTAIL))

932 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/bignum/%.c
933         @($(LHEAD) $(LINT.c) $< $(LTAIL))

935 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/acl/%.c
936         @($(LHEAD) $(LINT.c) $< $(LTAIL))

938 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/avl/%.c
939         @($(LHEAD) $(LINT.c) $< $(LTAIL))

941 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/brand/sn1/%.c
942         @($(LHEAD) $(LINT.c) $< $(LTAIL))

944 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/c2/%.c
945         @($(LHEAD) $(LINT.c) $< $(LTAIL))

947 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/conf/%.c
948         @($(LHEAD) $(LINT.c)    $< $(LTAIL))

950 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/contract/%.c
951         @($(LHEAD) $(LINT.c) $< $(LTAIL))

953 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/cpr/%.c
954         @($(LHEAD) $(LINT.c) $< $(LTAIL))

956 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/ctf/%.c
957         @($(LHEAD) $(LINT.c) $< $(LTAIL))

959 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/ctf/%.c
960         @($(LHEAD) $(LINT.c) $< $(LTAIL))

962 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/pci/%.c
963         @($(LHEAD) $(LINT.c) $< $(LTAIL))

965 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/devid/%.c
966         @($(LHEAD) $(LINT.c) $< $(LTAIL))

968 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/des/%.c
969         @($(LHEAD) $(LINT.c) $< $(LTAIL))

971 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/smbios/%.c
972         @($(LHEAD) $(LINT.c) $< $(LTAIL))

974 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/des/%.c
975         @($(LHEAD) $(LINT.c) $< $(LTAIL))

977 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/crypto/api/%.c
978         @($(LHEAD) $(LINT.c) $< $(LTAIL))

980 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/crypto/core/%.c
```



```
981         @($(LHEAD) $(LINT.c) $< $(LTAIL))

983 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/crypto/io/%.c
984         @($(LHEAD) $(LINT.c) $< $(LTAIL))

986 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/crypto/spi/%.c
987         @($(LHEAD) $(LINT.c) $< $(LTAIL))

989 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/disp/%.c
990         @($(LHEAD) $(LINT.c) $< $(LTAIL))

992 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/dtrace/%.c
993         @($(LHEAD) $(LINT.c) $< $(LTAIL))

995 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/exacct/%.c
996         @($(LHEAD) $(LINT.c) $< $(LTAIL))

998 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/exec/aout/%.c
999         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1001 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/exec/elf/%.c
1002         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1004 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/exec/intp/%.c
1005         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1007 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/exec/java/%.c
1008         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1010 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/%.c
1011         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1013 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/autofs/%.c
1014         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1016 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/cachefs/%.c
1017         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1019 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/ctfs/%.c
1020         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1022 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/doorfs/%.c
1023         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1025 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/devfs/%.c
1026         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1028 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/dev/%.c
1029         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1031 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/fd/%.c
1032         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1034 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/fifofs/%.c
1035         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1037 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/hsfs/%.c
1038         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1040 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/lofs/%.c
1041         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1043 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/mntfs/%.c
1044         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1046 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/fs/namefs/%.c
```

```
1047         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1049 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/nfs/%.c
1050         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1052 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/objfs/%.c
1053         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1055 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/pcfs/%.c
1056         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1058 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/portfs/%.c
1059         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1061 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/proc/%.c
1062         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1064 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/sockfs/%.c
1065         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1067 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/specfs/%.c
1068         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1070 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/swapfs/%.c
1071         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1073 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/tmpfs/%.c
1074         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1076 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/udfs/%.c
1077         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1079 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/ufs/%.c
1080         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1082 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/ufs_log/%.c
1083         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1085 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/zfs/%.c
1086         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1088 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/zfs/%.c
1089         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1091 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/gssapi/%.c
1092         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1094 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/gssapi/mechs/dummy/%.c
1095         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1097 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/%.c
1098         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1100 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/%.c
1101         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1103 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/des/%.c
1104         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1106 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/dk/%.c
1107         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1109 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/os/%.c
1110         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1112 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/arcfour/%.c
```



```
1113         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1115 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/enc_provider/%.c
1116         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1118 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/hash_provider/%.c
1119         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1121 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/keyhash_provider/%.c
1122         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1124 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/raw/%.c
1125         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1127 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/crc32/%.c
1128         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1130 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/old/%.c
1131         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1133 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/krb5/krb/%.c
1134         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1136 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/krb5/os/%.c
1137         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1139 $(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/mech/%.c
1140         @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

1142 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/%.c
1143         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1145 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/arp/%.c
1146         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1148 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/ip/%.c
1149         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1151 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/ipf/%.c
1152         @($(LHEAD) $(LINT.c) $(IPFFLAGS) $< $(LTAIL))

1154 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/pfil/%.c
1155         @($(LHEAD) $(LINT.c) $(IPFFLAG2) $< $(LTAIL))

1157 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/kssl/%.c
1158         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1160 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/net/patricia/%.c
1161         @($(LHEAD) $(LINT.c) $(IPFFLAGS) $< $(LTAIL))

1163 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/udp/%.c
1164         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1166 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/sctp/%.c
1167         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1169 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/tcp/%.c
1170         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1172 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/nca/%.c
1173         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1175 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/vni/%.c
1176         @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/1394/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/1394/adapters/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/1394/targets/av1394/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/1394/targets/dcam1394/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/1394/targets/scsa1394/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sbp2/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ath/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/legacy/diaudio/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/sada/amsrc2/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/sada/mixer/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/sada/framework/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/sada/drv/audio810/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/sada/drv/audiohd/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/sada/drv/audioixp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/sada/drv/audiots/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/bge/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/cardbus/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/dld/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/dls/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/drm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/hotplug/hpcsvc/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/hotplug/pciehpc/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/hotplug/pcishpc/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/hotplug/pcihp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/i2o/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/rds/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/ibd/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/sdp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/mgt/ibcm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/mgt/ibdm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/mgt/ibmf/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/ibnex/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/ibtl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/kb8042/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/kbtrans/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/aggr/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/hotspares/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/md/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/mirror/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/raid/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/softpart/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/stripe/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/notify/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/trans/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/mac/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/mac/plugins/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/net80211/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/pci-ide/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/pcmcia/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ppp/sppp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ppp/spppasyn/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ppp/spppcomp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ppp/spppptun/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/rsm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/rge/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sata/adapters/ahci/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sata/adapters/si3124/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sata/impl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/adapters/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/conf/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/impl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/targets/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/audio/usb_ac/%.
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/audio/usb_as/%.
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/audio/usb_ah/%.
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbskel/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/video/usbvc/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/hid/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/hidparser/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbkbm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbms/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/ugen/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/printer/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbser/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbser/usbsacm/
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbser/usbser_k
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbser/usbsprl/
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/hcd/openhci/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/hcd/ehci/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/hcd/uhci/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/hubd/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/scsa2usb/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/usb_mid/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/usb_ia/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/usba/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/usba10/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/vuidmice/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/chxge/com/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/chxge/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ixgb/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/xge/drv/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/xge/hal/xgehal/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/ipp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/ipp/ipgpc/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/ipp/dlcosmk/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/ipp/flowacct/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/ipp/dscpmk/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/ipp/meters/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/kmdb/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/krtld/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/ktli/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/lvm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/md5/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/net/dhcp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/nvpair/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/os/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/rpc/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/pcmcia/cs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/pcmcia/cis/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/pcmcia/nexus/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/pcmcia/pcs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/pcmcia/pem/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/rpc/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/rpc/sec/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/rpc/sec_gss/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/sha1/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/sha2/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/syscall/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/tnf/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/tsol/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/util/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/vm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/zmod/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

**********
1650 Wed Jan 31 21:21:16 2007
new/usr/src/uts/common/fs/zfs/gzip.c
1000002 gzip compression for ZFS
**********

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "@(#)gzip.c     1.2     07/01/31 SMI"

#include <sys/debug.h>
#include <sys/types.h>
#include <sys/zmod.h>

#ifdef _KERNEL
#include <sys/systm.h>
#else
#include <strings.h>
#endif

size_t
gzip_compress(void *s_start, void *d_start, size_t s_len, size_t d_len)
{
        size_t dstlen = d_len;

        ASSERT(d_len <= s_len);

        if (z_compress(d_start, &dstlen, s_start, s_len) != Z_OK) {
                if (d_len != s_len)
                        return (s_len);

                bcopy(s_start, d_start, s_len);
                return (s_len);
        }

        return (dstlen);
}

int
gzip_decompress(void *s_start, void *d_start, size_t s_len, size_t d_len)
{
        size_t dstlen = d_len;
```



```
        ASSERT(d_len >= s_len);

        if (z_uncompress(d_start, &dstlen, s_start, s_len) != Z_OK)
                return (-1);

        return (0);
}
```

```
**********************************************************
   10567 Wed Jan 31 21:21:17 2007
new/usr/src/uts/common/fs/zfs/sys/zio.h
1000002 gzip compression for ZFS
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _ZIO_H
#define _ZIO_H

#pragma ident   "@(#)zio.h      1.11    07/01/31 SMI"
#pragma ident   "@(#)zio.h      1.10    07/01/22 SMI"

#include <sys/zfs_context.h>
#include <sys/spa.h>
#include <sys/txg.h>
#include <sys/avl.h>
#include <sys/dkio.h>
#include <sys/fs/zfs.h>
#include <sys/zio_impl.h>

#ifdef  __cplusplus
extern "C" {
#endif

#define ZBT_MAGIC       0x210da7ab10c7a11ULL    /* zio data bloc tail */

typedef struct zio_block_tail {
        uint64_t        zbt_magic;      /* for validation, endianness   */
        zio_cksum_t     zbt_cksum;      /* 256-bit checksum             */
} zio_block_tail_t;
```

*unchanged_portion_omitted_*

```
#define ZIO_CHECKSUM_ON_VALUE   ZIO_CHECKSUM_FLETCHER_2
#define ZIO_CHECKSUM_DEFAULT    ZIO_CHECKSUM_ON

enum zio_compress {
        ZIO_COMPRESS_INHERIT = 0,
        ZIO_COMPRESS_ON,
        ZIO_COMPRESS_OFF,
        ZIO_COMPRESS_LZJB,
        ZIO_COMPRESS_EMPTY,
```



```
        ZIO_COMPRESS_GZIP,
        ZIO_COMPRESS_FUNCTIONS
};
```

*unchanged_portion_omitted_*

**********
***2333 Wed Jan 31 21:21:17 2007***
***new/usr/src/uts/common/fs/zfs/sys/zio_compress.h***
***1000002 gzip compression for ZFS***
**********

```
 1 /*
 2  * CDDL HEADER START
 3  *
 4  * The contents of this file are subject to the terms of the
 5  * Common Development and Distribution License (the "License").
 6  * You may not use this file except in compliance with the License.
 5  * Common Development and Distribution License, Version 1.0 only
 6  * (the "License").  You may not use this file except in compliance
 7  * with the License.
 7  *
 8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 9  * or http://www.opensolaris.org/os/licensing.
10  * See the License for the specific language governing permissions
11  * and limitations under the License.
12  *
13  * When distributing Covered Code, include this CDDL HEADER in each
14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15  * If applicable, add the following below this CDDL HEADER, with the
16  * fields enclosed by brackets "[]" replaced with your own identifying
17  * information: Portions Copyright [yyyy] [name of copyright owner]
18  *
19  * CDDL HEADER END
20  */

22 /*
23  * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
23  * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
24  * Use is subject to license terms.
25  */

27 #ifndef _SYS_ZIO_COMPRESS_H
28 #define _SYS_ZIO_COMPRESS_H

30 #pragma ident   "@(#)zio_compress.h     1.2     07/01/31 SMI"
30 #pragma ident   "@(#)zio_compress.h     1.1     05/10/30 SMI"

32 #include <sys/zio.h>

34 #ifdef  __cplusplus
35 extern "C" {
36 #endif

38 /*
39  * Common signature for all zio compress/decompress functions.
40  */
41 typedef size_t zio_compress_func_t(void *src, void *dst,
42     size_t s_len, size_t d_len);
43 typedef int zio_decompress_func_t(void *src, void *dst,
44     size_t s_len, size_t d_len);

46 /*
47  * Information about each compression function.
48  */
49 typedef struct zio_compress_info {
50         zio_compress_func_t     *ci_compress;
51         zio_decompress_func_t   *ci_decompress;
52         char                    *ci_name;
53 } zio_compress_info_t;

55 extern zio_compress_info_t zio_compress_table[ZIO_COMPRESS_FUNCTIONS];
```



```
57 /*
58  * Compression routines.
59  */
60 extern size_t lzjb_compress(void *src, void *dst, size_t s_len, size_t d_len);
61 extern int lzjb_decompress(void *src, void *dst, size_t s_len, size_t d_len);
62 extern size_t gzip_compress(void *src, void *dst, size_t s_len, size_t d_len);
63 extern int gzip_decompress(void *src, void *dst, size_t s_len, size_t d_len);

65 /*
66  * Compress and decompress data if necessary.
67  */
68 extern int zio_compress_data(int cpfunc, void *src, uint64_t srcsize,
69     void **destp, uint64_t *destsizep, uint64_t *destbufsizep);
70 extern int zio_decompress_data(int cpfunc, void *src, uint64_t srcsize,
71     void *dest, uint64_t destsize);

73 #ifdef  __cplusplus
74 }
```

***unchanged_portion_omitted_***

```
**********************************************************
    3662 Wed Jan 31 21:21:17 2007
new/usr/src/uts/common/fs/zfs/zio_compress.c
1000002 gzip compression for ZFS
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "@(#)zio_compress.c     1.4     07/01/31 SMI"
#pragma ident   "@(#)zio_compress.c     1.3     06/10/26 SMI"

#include <sys/zfs_context.h>
#include <sys/compress.h>
#include <sys/spa.h>
#include <sys/zio.h>
#include <sys/zio_compress.h>

/*
 * Compression vectors.
 */

zio_compress_info_t zio_compress_table[ZIO_COMPRESS_FUNCTIONS] = {
        {NULL,                  NULL,                   "inherit"},
        {NULL,                  NULL,                   "on"},
        {NULL,                  NULL,                   "uncompressed"},
        {lzjb_compress,         lzjb_decompress,        "lzjb"},
        {NULL,                  NULL,                   "empty"},
        {gzip_compress,         gzip_decompress,        "gzip"},
};
```

***unchanged_portion_omitted_***

```
**********************************************************
    1765 Wed Jan 31 21:21:17 2007
new/usr/src/uts/common/sys/zmod.h
1000001 add zlib compression support
**********************************************************
```

```
 1 /*
 2  * CDDL HEADER START
 3  *
 4  * The contents of this file are subject to the terms of the
 5  * Common Development and Distribution License (the "License").
 6  * You may not use this file except in compliance with the License.
 5  * Common Development and Distribution License, Version 1.0 only
 6  * (the "License").  You may not use this file except in compliance
 7  * with the License.
 7  *
 8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 9  * or http://www.opensolaris.org/os/licensing.
10  * See the License for the specific language governing permissions
11  * and limitations under the License.
12  *
13  * When distributing Covered Code, include this CDDL HEADER in each
14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15  * If applicable, add the following below this CDDL HEADER, with the
16  * fields enclosed by brackets "[]" replaced with your own identifying
17  * information: Portions Copyright [yyyy] [name of copyright owner]
18  *
19  * CDDL HEADER END
20  */

22 /*
23  * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
24  * Copyright 2003 Sun Microsystems, Inc.  All rights reserved.
24  * Use is subject to license terms.
25  */

27 #ifndef _ZMOD_H
28 #define _ZMOD_H

30 #pragma ident   "@(#)zmod.h     1.3     07/01/31 SMI"
31 #pragma ident   "@(#)zmod.h     1.2     05/07/18 SMI"

32 #ifdef  __cplusplus
33 extern "C" {
34 #endif

36 /*
37  * zmod - RFC-1950-compatible decompression routines
38  *
39  * This file provides the public interfaces to zmod, an in-kernel RFC 1950
40  * decompression library.  More information about the implementation of these
41  * interfaces can be found in the usr/src/uts/common/zmod/ directory.
42  */

44 #define Z_OK            0
45 #define Z_STREAM_END    1
46 #define Z_NEED_DICT     2
47 #define Z_ERRNO         (-1)
48 #define Z_STREAM_ERROR  (-2)
49 #define Z_DATA_ERROR    (-3)
50 #define Z_MEM_ERROR     (-4)
51 #define Z_BUF_ERROR     (-5)
52 #define Z_VERSION_ERROR (-6)

54 extern int z_uncompress(void *, size_t *, const void *, size_t);
55 extern int z_compress(void *, size_t *, const void *, size_t);
56 extern const char *z_strerror(int);
```



```
58 #ifdef  __cplusplus
59 }
_____unchanged_portion_omitted_
```

```
**********************************************************
   47873 Wed Jan 31 21:21:18 2007
new/usr/src/uts/common/zmod/deflate.c
1000001 add zlib compression support
**********************************************************
```

```
/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "@(#)deflate.c  1.2     07/01/31 SMI"

/*
 * deflate.c -- compress data using the deflation algorithm
 * Copyright (C) 1995-1998 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 *  ALGORITHM
 *
 *      The "deflation" process depends on being able to identify portions
 *      of the input text which are identical to earlier input (within a
 *      sliding window trailing behind the input currently being processed).
 *
 *      The most straightforward technique turns out to be the fastest for
 *      most input files: try all possible matches and select the longest.
 *      The key feature of this algorithm is that insertions into the string
 *      dictionary are very simple and thus fast, and deletions are avoided
 *      completely. Insertions are performed at each input character, whereas
 *      string matches are performed only when the previous match ends. So it
 *      is preferable to spend more time in matches to allow very fast string
 *      insertions and avoid deletions. The matching algorithm for small
 *      strings is inspired from that of Rabin & Karp. A brute force approach
 *      is used to find longer strings when a small match has been found.
 *      A similar algorithm is used in comic (by Jan-Mark Wams) and freeze
 *      (by Leonid Broukhis).
 *         A previous version of this file used a more sophisticated algorithm
 *      (by Fiala and Greene) which is guaranteed to run in linear amortized
 *      time, but has a larger average cost, uses more memory and is patented.
 *      However the F&G algorithm may be faster for some highly redundant
 *      files if the parameter max_chain_length (described below) is too large.
 *
 *  ACKNOWLEDGEMENTS
 *
 *      The idea of lazy evaluation of matches is due to Jan-Mark Wams, and
 *      I found it in 'freeze' written by Leonid Broukhis.
 *      Thanks to many people for bug reports and testing.
 *
 *  REFERENCES
 *
 *      Deutsch, L.P.,"DEFLATE Compressed Data Format Specification".
 *      Available in ftp://ds.internic.net/rfc/rfc1951.txt
 *
 *      A description of the Rabin and Karp algorithm is given in the book
 *         "Algorithms" by R. Sedgewick, Addison-Wesley, p252.
 *
 *      Fiala,E.R., and Greene,D.H.
 *         Data Compression with Finite Windows, Comm.ACM, 32,4 (1989) 490-595
 *
 */

#include "deflate.h"

/* ===========================================================================
 *  Function prototypes.
```



```
 */
typedef enum {
    need_more,      /* block not completed, need more input or more output */
    block_done,     /* block flush performed */
    finish_started, /* finish started, need only more output at next deflate */
    finish_done     /* finish done, accept no more input or output */
} block_state;

typedef block_state (*compress_func) OF((deflate_state *s, int flush));
/* Compression function. Returns the block state after the call. */

local void fill_window    OF((deflate_state *s));
local block_state deflate_stored OF((deflate_state *s, int flush));
local block_state deflate_fast   OF((deflate_state *s, int flush));
local block_state deflate_slow   OF((deflate_state *s, int flush));
local void lm_init        OF((deflate_state *s));
local void putShortMSB    OF((deflate_state *s, uInt b));
local void flush_pending  OF((z_streamp strm));
local int read_buf        OF((z_streamp strm, Bytef *buf, unsigned size));
#ifdef ASMV
      void match_init OF((void)); /* asm code initialization */
      uInt longest_match  OF((deflate_state *s, IPos cur_match));
#else
local uInt longest_match  OF((deflate_state *s, IPos cur_match));
#endif

#ifdef DEBUG
local  void check_match OF((deflate_state *s, IPos start, IPos match,
                            int length));
#endif

/* ===========================================================================
 * Local data
 */

#define NIL 0
/* Tail of hash chains */

#ifndef TOO_FAR
#  define TOO_FAR 4096
#endif
/* Matches of length 3 are discarded if their distance exceeds TOO_FAR */

#define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)
/* Minimum amount of lookahead, except at the end of the input file.
 * See deflate.c for comments about the MIN_MATCH+1.
 */

/* Values for max_lazy_match, good_match and max_chain_length, depending on
 * the desired pack level (0..9). The values given below have been tuned to
 * exclude worst case performance for pathological files. Better values may be
 * found for specific files.
 */
typedef struct config_s {
   ush good_length; /* reduce lazy search above this match length */
   ush max_lazy;    /* do not perform lazy search above this match length */
   ush nice_length; /* quit search above this match length */
   ush max_chain;
   compress_func func;
} config;

local const config configuration_table[10] = {
/*      good lazy nice chain */
/* 0 */ {0,    0,  0,    0, deflate_stored},  /* store only */
/* 1 */ {4,    4,  8,    4, deflate_fast}, /* maximum speed, no lazy matches */
/* 2 */ {4,    5, 16,    8, deflate_fast},
```

```
/* 3 */ {4,    6, 32,   32, deflate_fast},

/* 4 */ {4,    4, 16,   16, deflate_slow},  /* lazy matches */
/* 5 */ {8,   16, 32,   32, deflate_slow},
/* 6 */ {8,   16, 128, 128, deflate_slow},
/* 7 */ {8,   32, 128, 256, deflate_slow},
/* 8 */ {32, 128, 258, 1024, deflate_slow},
/* 9 */ {32, 258, 258, 4096, deflate_slow}}; /* maximum compression */

/* Note: the deflate() code requires max_lazy >= MIN_MATCH and max_chain >= 4
 * For deflate_fast() (levels <= 3) good is ignored and lazy has a different
 * meaning.
 */

#define EQUAL 0
/* result of memcmp for equal strings */

struct static_tree_desc_s {int dummy;}; /* for buggy compilers */

/* ===========================================================================
 * Update a hash value with the given input byte
 * IN  assertion: all calls to to UPDATE_HASH are made with consecutive
 *    input characters, so that a running hash key can be computed from the
 *    previous key instead of complete recalculation each time.
 */
#define UPDATE_HASH(s,h,c) (h = (((h)<<s->hash_shift) ^ (c)) & s->hash_mask)


/* ===========================================================================
 * Insert string str in the dictionary and set match_head to the previous head
 * of the hash chain (the most recent string with same hash key). Return
 * the previous length of the hash chain.
 * If this file is compiled with -DFASTEST, the compression level is forced
 * to 1, and no hash chains are maintained.
 * IN  assertion: all calls to to INSERT_STRING are made with consecutive
 *    input characters and the first MIN_MATCH bytes of str are valid
 *    (except for the last MIN_MATCH-1 bytes of the input file).
 */
#ifdef FASTEST
#define INSERT_STRING(s, str, match_head) \
   (UPDATE_HASH(s, s->ins_h, s->window[(str) + (MIN_MATCH-1)]), \
    match_head = s->head[s->ins_h], \
    s->head[s->ins_h] = (Pos)(str))
#else
#define INSERT_STRING(s, str, match_head) \
   (UPDATE_HASH(s, s->ins_h, s->window[(str) + (MIN_MATCH-1)]), \
    s->prev[(str) & s->w_mask] = match_head = s->head[s->ins_h], \
    s->head[s->ins_h] = (Pos)(str))
#endif

/* ===========================================================================
 * Initialize the hash table (avoiding 64K overflow for 16 bit systems).
 * prev[] will be initialized on the fly.
 */
#define CLEAR_HASH(s) \
    s->head[s->hash_size-1] = NIL; \
    zmemzero((Bytef *)s->head, (unsigned)(s->hash_size-1)*sizeof(*s->head));

/* ========================================================================= */
int ZEXPORT deflateInit_(strm, level, version, stream_size)
    z_streamp strm;
    int level;
    const char *version;
    int stream_size;
{
    return deflateInit2_(strm, level, Z_DEFLATED, MAX_WBITS, DEF_MEM_LEVEL,
```



```
                         Z_DEFAULT_STRATEGY, version, stream_size);
    /* To do: ignore strm->next_in if we use it as window */
}

/* ========================================================================= */
int ZEXPORT deflateInit2_(strm, level, method, windowBits, memLevel, strategy,
                  version, stream_size)
    z_streamp strm;
    int  level;
    int  method;
    int  windowBits;
    int  memLevel;
    int  strategy;
    const char *version;
    int stream_size;
{
    deflate_state *s;
    int noheader = 0;
    static const char* my_version = ZLIB_VERSION;

    ushf *overlay;
    /* We overlay pending_buf and d_buf+l_buf. This works since the average
     * output size for (length,distance) codes is <= 24 bits.
     */

    if (version == Z_NULL || version[0] != my_version[0] ||
        stream_size != sizeof(z_stream)) {
        return Z_VERSION_ERROR;
    }
    if (strm == Z_NULL) return Z_STREAM_ERROR;

    strm->msg = Z_NULL;
    if (strm->zalloc == Z_NULL) {
        strm->zalloc = zcalloc;
        strm->opaque = (voidpf)0;
    }
    if (strm->zfree == Z_NULL) strm->zfree = zcfree;

    if (level == Z_DEFAULT_COMPRESSION) level = 6;
#ifdef FASTEST
    level = 1;
#endif

    if (windowBits < 0) { /* undocumented feature: suppress zlib header */
        noheader = 1;
        windowBits = -windowBits;
    }
    if (memLevel < 1 || memLevel > MAX_MEM_LEVEL || method != Z_DEFLATED ||
        windowBits < 8 || windowBits > 15 || level < 0 || level > 9 ||
        strategy < 0 || strategy > Z_HUFFMAN_ONLY) {
        return Z_STREAM_ERROR;
    }
    s = (deflate_state *) ZALLOC(strm, 1, sizeof(deflate_state));
    if (s == Z_NULL) return Z_MEM_ERROR;
    strm->state = (struct internal_state FAR *)s;
    s->strm = strm;

    s->noheader = noheader;
    s->w_bits = windowBits;
    s->w_size = 1 << s->w_bits;
    s->w_mask = s->w_size - 1;

    s->hash_bits = memLevel + 7;
    s->hash_size = 1 << s->hash_bits;
    s->hash_mask = s->hash_size - 1;
    s->hash_shift =  ((s->hash_bits+MIN_MATCH-1)/MIN_MATCH);
```

```
261     s->window = (Bytef *) ZALLOC(strm, s->w_size, 2*sizeof(Byte));
262     s->prev   = (Posf *)  ZALLOC(strm, s->w_size, sizeof(Pos));
263     s->head   = (Posf *)  ZALLOC(strm, s->hash_size, sizeof(Pos));

265     s->lit_bufsize = 1 << (memLevel + 6); /* 16K elements by default */

267     overlay = (ushf *) ZALLOC(strm, s->lit_bufsize, sizeof(ush)+2);
268     s->pending_buf = (uchf *) overlay;
269     s->pending_buf_size = (ulg)s->lit_bufsize * (sizeof(ush)+2L);

271     if (s->window == Z_NULL || s->prev == Z_NULL || s->head == Z_NULL ||
272         s->pending_buf == Z_NULL) {
273         deflateEnd (strm);
274         return Z_MEM_ERROR;
275     }
276     s->d_buf = overlay + s->lit_bufsize/sizeof(ush);
277     s->l_buf = s->pending_buf + (1+sizeof(ush))*s->lit_bufsize;

279     s->level = level;
280     s->strategy = strategy;
281     s->method = (Byte)method;

283     return deflateReset(strm);
284 }

286 /* ========================================================================= */
287 int ZEXPORT deflateSetDictionary (strm, dictionary, dictLength)
288     z_streamp strm;
289     const Bytef *dictionary;
290     uInt  dictLength;
291 {
292     deflate_state *s;
293     uInt length = dictLength;
294     uInt n;
295     IPos hash_head = 0;

297     if (strm == Z_NULL || strm->state == Z_NULL || dictionary == Z_NULL ||
298         strm->state->status != INIT_STATE) return Z_STREAM_ERROR;

300     s = strm->state;
301     strm->adler = adler32(strm->adler, dictionary, dictLength);

303     if (length < MIN_MATCH) return Z_OK;
304     if (length > MAX_DIST(s)) {
305         length = MAX_DIST(s);
306 #ifndef USE_DICT_HEAD
307         dictionary += dictLength - length; /* use the tail of the dictionary */
308 #endif
309     }
310     zmemcpy(s->window, dictionary, length);
311     s->strstart = length;
312     s->block_start = (long)length;

314     /* Insert all strings in the hash table (except for the last two bytes).
315      * s->lookahead stays null, so s->ins_h will be recomputed at the next
316      * call of fill_window.
317      */
318     s->ins_h = s->window[0];
319     UPDATE_HASH(s, s->ins_h, s->window[1]);
320     for (n = 0; n <= length - MIN_MATCH; n++) {
321         INSERT_STRING(s, n, hash_head);
322     }
323     if (hash_head) hash_head = 0;  /* to make compiler happy */
324     return Z_OK;
325 }
```



```
327 /* ========================================================================= */
328 int ZEXPORT deflateReset (strm)
329     z_streamp strm;
330 {
331     deflate_state *s;
332
333     if (strm == Z_NULL || strm->state == Z_NULL ||
334         strm->zalloc == Z_NULL || strm->zfree == Z_NULL) return Z_STREAM_ERROR;

336     strm->total_in = strm->total_out = 0;
337     strm->msg = Z_NULL; /* use zfree if we ever allocate msg dynamically */
338     strm->data_type = Z_UNKNOWN;

340     s = (deflate_state *)strm->state;
341     s->pending = 0;
342     s->pending_out = s->pending_buf;

344     if (s->noheader < 0) {
345         s->noheader = 0; /* was set to -1 by deflate(..., Z_FINISH); */
346     }
347     s->status = s->noheader ? BUSY_STATE : INIT_STATE;
348     strm->adler = 1;
349     s->last_flush = Z_NO_FLUSH;

351     _tr_init(s);
352     lm_init(s);

354     return Z_OK;
355 }

357 /* ========================================================================= */
358 int ZEXPORT deflateParams(strm, level, strategy)
359     z_streamp strm;
360     int level;
361     int strategy;
362 {
363     deflate_state *s;
364     compress_func func;
365     int err = Z_OK;

367     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
368     s = strm->state;

370     if (level == Z_DEFAULT_COMPRESSION) {
371         level = 6;
372     }
373     if (level < 0 || level > 9 || strategy < 0 || strategy > Z_HUFFMAN_ONLY) {
374         return Z_STREAM_ERROR;
375     }
376     func = configuration_table[s->level].func;

378     if (func != configuration_table[level].func && strm->total_in != 0) {
379         /* Flush the last buffer: */
380         err = deflate(strm, Z_PARTIAL_FLUSH);
381     }
382     if (s->level != level) {
383         s->level = level;
384         s->max_lazy_match   = configuration_table[level].max_lazy;
385         s->good_match       = configuration_table[level].good_length;
386         s->nice_match       = configuration_table[level].nice_length;
387         s->max_chain_length = configuration_table[level].max_chain;
388     }
389     s->strategy = strategy;
390     return err;
391 }
```

```
/* ========================================================================
 * Put a short in the pending buffer. The 16-bit value is put in MSB order.
 * IN assertion: the stream state is correct and there is enough room in
 * pending_buf.
 */
local void putShortMSB (s, b)
    deflate_state *s;
    uInt b;
{
    put_byte(s, (Byte)(b >> 8));
    put_byte(s, (Byte)(b & 0xff));
}

/* =========================================================================
 * Flush as much pending output as possible. All deflate() output goes
 * through this function so some applications may wish to modify it
 * to avoid allocating a large strm->next_out buffer and copying into it.
 * (See also read_buf()).
 */
local void flush_pending(strm)
    z_streamp strm;
{
    unsigned len = strm->state->pending;

    if (len > strm->avail_out) len = strm->avail_out;
    if (len == 0) return;

    zmemcpy(strm->next_out, strm->state->pending_out, len);
    strm->next_out  += len;
    strm->state->pending_out  += len;
    strm->total_out += len;
    strm->avail_out  -= len;
    strm->state->pending -= len;
    if (strm->state->pending == 0) {
        strm->state->pending_out = strm->state->pending_buf;
    }
}

/* ========================================================================= */
int ZEXPORT deflate (strm, flush)
    z_streamp strm;
    int flush;
{
    int old_flush; /* value of flush param for previous deflate call */
    deflate_state *s;

    if (strm == Z_NULL || strm->state == Z_NULL ||
        flush > Z_FINISH || flush < 0) {
        return Z_STREAM_ERROR;
    }
    s = strm->state;

    if (strm->next_out == Z_NULL ||
        (strm->next_in == Z_NULL && strm->avail_in != 0) ||
        (s->status == FINISH_STATE && flush != Z_FINISH)) {
        return Z_STREAM_ERROR;
    }
    if (strm->avail_out == 0) return Z_BUF_ERROR;

    s->strm = strm; /* just in case */
    old_flush = s->last_flush;
    s->last_flush = flush;

    /* Write the zlib header */
    if (s->status == INIT_STATE) {
```



```
        uInt header = (Z_DEFLATED + ((s->w_bits-8)<<4)) << 8;
        uInt level_flags = (s->level-1) >> 1;

        if (level_flags > 3) level_flags = 3;
        header |= (level_flags << 6);
        if (s->strstart != 0) header |= PRESET_DICT;
        header += 31 - (header % 31);

        s->status = BUSY_STATE;
        putShortMSB(s, header);

        /* Save the adler32 of the preset dictionary: */
        if (s->strstart != 0) {
            putShortMSB(s, (uInt)(strm->adler >> 16));
            putShortMSB(s, (uInt)(strm->adler & 0xffff));
        }
        strm->adler = 1L;
    }

    /* Flush as much pending output as possible */
    if (s->pending != 0) {
        flush_pending(strm);
        if (strm->avail_out == 0) {
            /* Since avail_out is 0, deflate will be called again with
             * more output space, but possibly with both pending and
             * avail_in equal to zero. There won't be anything to do,
             * but this is not an error situation so make sure we
             * return OK instead of BUF_ERROR at next call of deflate:
             */
            s->last_flush = -1;
            return Z_OK;
        }

    /* Make sure there is something to do and avoid duplicate consecutive
     * flushes. For repeated and useless calls with Z_FINISH, we keep
     * returning Z_STREAM_END instead of Z_BUFF_ERROR.
     */
    } else if (strm->avail_in == 0 && flush <= old_flush &&
               flush != Z_FINISH) {
        return Z_BUF_ERROR;
    }

    /* User must not provide more input after the first FINISH: */
    if (s->status == FINISH_STATE && strm->avail_in != 0) {
        return Z_BUF_ERROR;
    }

    /* Start a new block or continue the current one.
     */
    if (strm->avail_in != 0 || s->lookahead != 0 ||
        (flush != Z_NO_FLUSH && s->status != FINISH_STATE)) {
        block_state bstate;

        bstate = (*(configuration_table[s->level].func))(s, flush);

        if (bstate == finish_started || bstate == finish_done) {
            s->status = FINISH_STATE;
        }
        if (bstate == need_more || bstate == finish_started) {
            if (strm->avail_out == 0) {
                s->last_flush = -1; /* avoid BUF_ERROR next call, see above */
            }
            return Z_OK;
            /* If flush != Z_NO_FLUSH && avail_out == 0, the next call
             * of deflate should use the same flush parameter to make sure
```

```
524              * that the flush is complete. So we don't have to output an
525              * empty block here, this will be done at next call. This also
526              * ensures that for a very small output buffer, we emit at most
527              * one empty block.
528              */
529         }
530         if (bstate == block_done) {
531             if (flush == Z_PARTIAL_FLUSH) {
532                 _tr_align(s);
533             } else { /* FULL_FLUSH or SYNC_FLUSH */
534                 _tr_stored_block(s, (char*)0, 0L, 0);
535                 /* For a full flush, this empty block will be recognized
536                  * as a special marker by inflate_sync().
537                  */
538                 if (flush == Z_FULL_FLUSH) {
539                     CLEAR_HASH(s);             /* forget history */
540                 }
541             }
542             flush_pending(strm);
543             if (strm->avail_out == 0) {
544               s->last_flush = -1; /* avoid BUF_ERROR at next call, see above */
545               return Z_OK;
546             }
547         }
548     }
549     Assert(strm->avail_out > 0, "bug2");
550
551     if (flush != Z_FINISH) return Z_OK;
552     if (s->noheader) return Z_STREAM_END;
553
554     /* Write the zlib trailer (adler32) */
555     putShortMSB(s, (uInt)(strm->adler >> 16));
556     putShortMSB(s, (uInt)(strm->adler & 0xffff));
557     flush_pending(strm);
558     /* If avail_out is zero, the application will call deflate again
559      * to flush the rest.
560      */
561     s->noheader = -1; /* write the trailer only once! */
562     return s->pending != 0 ? Z_OK : Z_STREAM_END;
563 }
564
565 /* ========================================================================= */
566 int ZEXPORT deflateEnd (strm)
567     z_streamp strm;
568 {
569     int status;
570
571     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
572
573     status = strm->state->status;
574     if (status != INIT_STATE && status != BUSY_STATE &&
575         status != FINISH_STATE) {
576       return Z_STREAM_ERROR;
577     }
578
579     /* Deallocate in reverse order of allocations: */
580     TRY_FREE(strm, strm->state->pending_buf);
581     TRY_FREE(strm, strm->state->head);
582     TRY_FREE(strm, strm->state->prev);
583     TRY_FREE(strm, strm->state->window);
584
585     ZFREE(strm, strm->state);
586     strm->state = Z_NULL;
587
588     return status == BUSY_STATE ? Z_DATA_ERROR : Z_OK;
589 }
```



```
590
591 /* ========================================================================
592  * Copy the source state to the destination state.
593  * To simplify the source, this is not supported for 16-bit MSDOS (which
594  * doesn't have enough memory anyway to duplicate compression states).
595  */
596 int ZEXPORT deflateCopy (dest, source)
597     z_streamp dest;
598     z_streamp source;
599 {
600 #ifdef MAXSEG_64K
601     return Z_STREAM_ERROR;
602 #else
603     deflate_state *ds;
604     deflate_state *ss;
605     ushf *overlay;
606
607
608     if (source == Z_NULL || dest == Z_NULL || source->state == Z_NULL) {
609         return Z_STREAM_ERROR;
610     }
611
612     ss = source->state;
613
614     *dest = *source;
615
616     ds = (deflate_state *) ZALLOC(dest, 1, sizeof(deflate_state));
617     if (ds == Z_NULL) return Z_MEM_ERROR;
618     dest->state = (struct internal_state FAR *) ds;
619     *ds = *ss;
620     ds->strm = dest;
621
622     ds->window = (Bytef *) ZALLOC(dest, ds->w_size, 2*sizeof(Byte));
623     ds->prev   = (Posf *)  ZALLOC(dest, ds->w_size, sizeof(Pos));
624     ds->head   = (Posf *)  ZALLOC(dest, ds->hash_size, sizeof(Pos));
625     overlay = (ushf *) ZALLOC(dest, ds->lit_bufsize, sizeof(ush)+2);
626     ds->pending_buf = (uchf *) overlay;
627
628     if (ds->window == Z_NULL || ds->prev == Z_NULL || ds->head == Z_NULL ||
629         ds->pending_buf == Z_NULL) {
630         deflateEnd (dest);
631         return Z_MEM_ERROR;
632     }
633     /* following zmemcpy do not work for 16-bit MSDOS */
634     zmemcpy(ds->window, ss->window, ds->w_size * 2 * sizeof(Byte));
635     zmemcpy((void *)ds->prev, (void *)ss->prev, ds->w_size * sizeof(Pos));
636     zmemcpy((void *)ds->head, (void *)ss->head, ds->hash_size * sizeof(Pos));
637     zmemcpy(ds->pending_buf, ss->pending_buf, (uInt)ds->pending_buf_size);
638
639     ds->pending_out = ds->pending_buf + (ss->pending_out - ss->pending_buf);
640     ds->d_buf = overlay + ds->lit_bufsize/sizeof(ush);
641     ds->l_buf = ds->pending_buf + (1+sizeof(ush))*ds->lit_bufsize;
642
643     ds->l_desc.dyn_tree = ds->dyn_ltree;
644     ds->d_desc.dyn_tree = ds->dyn_dtree;
645     ds->bl_desc.dyn_tree = ds->bl_tree;
646
647     return Z_OK;
648 #endif
649 }
650
651 /* ===========================================================================
652  * Read a new buffer from the current input stream, update the adler32
653  * and total number of bytes read.  All deflate() input goes through
654  * this function so some applications may wish to modify it to avoid
655  * allocating a large strm->next_in buffer and copying from it.
```

```
656  * (See also flush_pending()).
657  */
658 local int read_buf(strm, buf, size)
659     z_streamp strm;
660     Bytef *buf;
661     unsigned size;
662 {
663     unsigned len = strm->avail_in;

665     if (len > size) len = size;
666     if (len == 0) return 0;

668     strm->avail_in  -= len;

670     if (!strm->state->noheader) {
671         strm->adler = adler32(strm->adler, strm->next_in, len);
672     }
673     zmemcpy(buf, strm->next_in, len);
674     strm->next_in  += len;
675     strm->total_in += len;

677     return (int)len;
678 }

680 /* ===========================================================================
681  * Initialize the "longest match" routines for a new zlib stream
682  */
683 local void lm_init (s)
684     deflate_state *s;
685 {
686     s->window_size = (ulg)2L*s->w_size;

688     CLEAR_HASH(s);

690     /* Set the default configuration parameters:
691      */
692     s->max_lazy_match   = configuration_table[s->level].max_lazy;
693     s->good_match       = configuration_table[s->level].good_length;
694     s->nice_match       = configuration_table[s->level].nice_length;
695     s->max_chain_length = configuration_table[s->level].max_chain;

697     s->strstart = 0;
698     s->block_start = 0L;
699     s->lookahead = 0;
700     s->match_length = s->prev_length = MIN_MATCH-1;
701     s->match_available = 0;
702     s->ins_h = 0;
703 #ifdef ASMV
704     match_init(); /* initialize the asm code */
705 #endif
706 }

708 /* ===========================================================================
709  * Set match_start to the longest match starting at the given string and
710  * return its length. Matches shorter or equal to prev_length are discarded,
711  * in which case the result is equal to prev_length and match_start is
712  * garbage.
713  * IN assertions: cur_match is the head of the hash chain for the current
714  *   string (strstart) and its distance is <= MAX_DIST, and prev_length >= 1
715  * OUT assertion: the match length is not greater than s->lookahead.
716  */
717 #ifndef ASMV
718 /* For 80x86 and 680x0, an optimized version will be provided in match.asm or
719  * match.S. The code will be functionally equivalent.
720  */
721 #ifndef FASTEST
```



```
722 local uInt longest_match(s, cur_match)
723     deflate_state *s;
724     IPos cur_match;                             /* current match */
725 {
726     unsigned chain_length = s->max_chain_length;/* max hash chain length */
727     register Bytef *scan = s->window + s->strstart; /* current string */
728     register Bytef *match;                       /* matched string */
729     register int len;                           /* length of current match */
730     int best_len = s->prev_length;              /* best match length so far */
731     int nice_match = s->nice_match;             /* stop if match long enough */
732     IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
733         s->strstart - (IPos)MAX_DIST(s) : NIL;
734     /* Stop when cur_match becomes <= limit. To simplify the code,
735      * we prevent matches with the string of window index 0.
736      */
737     Posf *prev = s->prev;
738     uInt wmask = s->w_mask;

740 #ifdef UNALIGNED_OK
741     /* Compare two bytes at a time. Note: this is not always beneficial.
742      * Try with and without -DUNALIGNED_OK to check.
743      */
744     register Bytef *strend = s->window + s->strstart + MAX_MATCH - 1;
745     register ush scan_start = *(ushf*)scan;
746     register ush scan_end   = *(ushf*)(scan+best_len-1);
747 #else
748     register Bytef *strend = s->window + s->strstart + MAX_MATCH;
749     register Byte scan_end1  = scan[best_len-1];
750     register Byte scan_end   = scan[best_len];
751 #endif

753     /* The code is optimized for HASH_BITS >= 8 and MAX_MATCH-2 multiple of 16.
754      * It is easy to get rid of this optimization if necessary.
755      */
756     Assert(s->hash_bits >= 8 && MAX_MATCH == 258, "Code too clever");

758     /* Do not waste too much time if we already have a good match: */
759     if (s->prev_length >= s->good_match) {
760         chain_length >>= 2;
761     }
762     /* Do not look for matches beyond the end of the input. This is necessary
763      * to make deflate deterministic.
764      */
765     if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;

767     Assert((ulg)s->strstart <= s->window_size-MIN_LOOKAHEAD, "need lookahead");

769     do {
770         Assert(cur_match < s->strstart, "no future");
771         match = s->window + cur_match;

773         /* Skip to next match if the match length cannot increase
774          * or if the match length is less than 2:
775          */
776 #if (defined(UNALIGNED_OK) && MAX_MATCH == 258)
777         /* This code assumes sizeof(unsigned short) == 2. Do not use
778          * UNALIGNED_OK if your compiler uses a different size.
779          */
780         if (*(ushf*)(match+best_len-1) != scan_end ||
781             *(ushf*)match != scan_start) continue;

783         /* It is not necessary to compare scan[2] and match[2] since they are
784          * always equal when the other bytes match, given that the hash keys
785          * are equal and that HASH_BITS >= 8. Compare 2 bytes at a time at
786          * strstart+3, +5, ... up to strstart+257. We check for insufficient
787          * lookahead only every 4th comparison; the 128th check will be made
```

```
788          * at strstart+257. If MAX_MATCH-2 is not a multiple of 8, it is
789          * necessary to put more guard bytes at the end of the window, or
790          * to check more often for insufficient lookahead.
791          */
792         Assert(scan[2] == match[2], "scan[2]?");
793         scan++, match++;
794         do {
795         } while (*(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
796                  *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
797                  *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
798                  *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
799                  scan < strend);
800         /* The funny "do {}" generates better code on most compilers */
801
802         /* Here, scan <= window+strstart+257 */
803         Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");
804         if (*scan == *match) scan++;
805
806         len = (MAX_MATCH - 1) - (int)(strend-scan);
807         scan = strend - (MAX_MATCH-1);
808
809 #else /* UNALIGNED_OK */
810
811         if (match[best_len]   != scan_end  ||
812             match[best_len-1] != scan_end1 ||
813             *match            != *scan     ||
814             *++match          != scan[1])      continue;
815
816         /* The check at best_len-1 can be removed because it will be made
817          * again later. (This heuristic is not always a win.)
818          * It is not necessary to compare scan[2] and match[2] since they
819          * are always equal when the other bytes match, given that
820          * the hash keys are equal and that HASH_BITS >= 8.
821          */
822         scan += 2, match++;
823         Assert(*scan == *match, "match[2]?");
824
825         /* We check for insufficient lookahead only every 8th comparison;
826          * the 256th check will be made at strstart+258.
827          */
828         do {
829         } while (*++scan == *++match && *++scan == *++match &&
830                  *++scan == *++match && *++scan == *++match &&
831                  *++scan == *++match && *++scan == *++match &&
832                  *++scan == *++match && *++scan == *++match &&
833                  scan < strend);
834
835         Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");
836
837         len = MAX_MATCH - (int)(strend - scan);
838         scan = strend - MAX_MATCH;
839
840 #endif /* UNALIGNED_OK */
841
842         if (len > best_len) {
843             s->match_start = cur_match;
844             best_len = len;
845             if (len >= nice_match) break;
846 #ifdef UNALIGNED_OK
847             scan_end = *(ushf*)(scan+best_len-1);
848 #else
849             scan_end1  = scan[best_len-1];
850             scan_end   = scan[best_len];
851 #endif
852         }
853     } while ((cur_match = prev[cur_match & wmask]) > limit
```



```
854              && --chain_length != 0);
855
856     if ((uInt)best_len <= s->lookahead) return (uInt)best_len;
857     return s->lookahead;
858 }
859
860 #else /* FASTEST */
861 /* ---------------------------------------------------------------------------
862  * Optimized version for level == 1 only
863  */
864 local uInt longest_match(s, cur_match)
865     deflate_state *s;
866     IPos cur_match;                             /* current match */
867 {
868     register Bytef *scan = s->window + s->strstart; /* current string */
869     register Bytef *match;                       /* matched string */
870     register int len;                           /* length of current match */
871     register Bytef *strend = s->window + s->strstart + MAX_MATCH;
872
873     /* The code is optimized for HASH_BITS >= 8 and MAX_MATCH-2 multiple of 16.
874      * It is easy to get rid of this optimization if necessary.
875      */
876     Assert(s->hash_bits >= 8 && MAX_MATCH == 258, "Code too clever");
877
878     Assert((ulg)s->strstart <= s->window_size-MIN_LOOKAHEAD, "need lookahead");
879
880     Assert(cur_match < s->strstart, "no future");
881
882     match = s->window + cur_match;
883
884     /* Return failure if the match length is less than 2:
885      */
886     if (match[0] != scan[0] || match[1] != scan[1]) return MIN_MATCH-1;
887
888     /* The check at best_len-1 can be removed because it will be made
889      * again later. (This heuristic is not always a win.)
890      * It is not necessary to compare scan[2] and match[2] since they
891      * are always equal when the other bytes match, given that
892      * the hash keys are equal and that HASH_BITS >= 8.
893      */
894     scan += 2, match += 2;
895     Assert(*scan == *match, "match[2]?");
896
897     /* We check for insufficient lookahead only every 8th comparison;
898      * the 256th check will be made at strstart+258.
899      */
900     do {
901     } while (*++scan == *++match && *++scan == *++match &&
902              *++scan == *++match && *++scan == *++match &&
903              *++scan == *++match && *++scan == *++match &&
904              *++scan == *++match && *++scan == *++match &&
905              scan < strend);
906
907     Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");
908
909     len = MAX_MATCH - (int)(strend - scan);
910
911     if (len < MIN_MATCH) return MIN_MATCH - 1;
912
913     s->match_start = cur_match;
914     return len <= s->lookahead ? len : s->lookahead;
915 }
916 #endif /* FASTEST */
917 #endif /* ASMV */
918
919 #  define check_match(s, start, match, length)
```

```
/* ===========================================================================
 * Fill the window when the lookahead becomes insufficient.
 * Updates strstart and lookahead.
 *
 * IN assertion: lookahead < MIN_LOOKAHEAD
 * OUT assertions: strstart <= window_size-MIN_LOOKAHEAD
 *    At least one byte has been read, or avail_in == 0; reads are
 *    performed for at least two bytes (required for the zip translate_eol
 *    option -- not supported here).
 */
local void fill_window(s)
    deflate_state *s;
{
    register unsigned n, m;
    register Posf *p;
    unsigned more;    /* Amount of free space at the end of the window. */
    uInt wsize = s->w_size;

    do {
        more = (unsigned)(s->window_size -(ulg)s->lookahead -(ulg)s->strstart);

        /* Deal with !@#$% 64K limit: */
        if (more == 0 && s->strstart == 0 && s->lookahead == 0) {
            more = wsize;

        } else if (more == (unsigned)(-1)) {
            /* Very unlikely, but possible on 16 bit machine if strstart == 0
             * and lookahead == 1 (input done one byte at time)
             */
            more--;

        /* If the window is almost full and there is insufficient lookahead,
         * move the upper half to the lower one to make room in the upper half.
         */
        } else if (s->strstart >= wsize+MAX_DIST(s)) {

            zmemcpy(s->window, s->window+wsize, (unsigned)wsize);
            s->match_start -= wsize;
            s->strstart    -= wsize; /* we now have strstart >= MAX_DIST */
            s->block_start -= (long) wsize;

            /* Slide the hash table (could be avoided with 32 bit values
               at the expense of memory usage). We slide even when level == 0
               to keep the hash table consistent if we switch back to level > 0
               later. (Using level 0 permanently is not an optimal usage of
               zlib, so we don't care about this pathological case.)
             */
            n = s->hash_size;
            p = &s->head[n];
            do {
                m = *--p;
                *p = (Pos)(m >= wsize ? m-wsize : NIL);
            } while (--n);

            n = wsize;
#ifndef FASTEST
            p = &s->prev[n];
            do {
                m = *--p;
                *p = (Pos)(m >= wsize ? m-wsize : NIL);
                /* If n is not on any hash chain, prev[n] is garbage but
                 * its value will never be used.
                 */
            } while (--n);
#endif
```



```
            more += wsize;
        }
        if (s->strm->avail_in == 0) return;

        /* If there was no sliding:
         *    strstart <= WSIZE+MAX_DIST-1 && lookahead <= MIN_LOOKAHEAD - 1 &&
         *    more == window_size - lookahead - strstart
         * => more >= window_size - (MIN_LOOKAHEAD-1 + WSIZE + MAX_DIST-1)
         * => more >= window_size - 2*WSIZE + 2
         * In the BIG_MEM or MMAP case (not yet supported),
         *   window_size == input_size + MIN_LOOKAHEAD  &&
         *   strstart + s->lookahead <= input_size => more >= MIN_LOOKAHEAD.
         * Otherwise, window_size == 2*WSIZE so more >= 2.
         * If there was sliding, more >= WSIZE. So in all cases, more >= 2.
         */
        Assert(more >= 2, "more < 2");

        n = read_buf(s->strm, s->window + s->strstart + s->lookahead, more);
        s->lookahead += n;

        /* Initialize the hash value now that we have some input: */
        if (s->lookahead >= MIN_MATCH) {
            s->ins_h = s->window[s->strstart];
            UPDATE_HASH(s, s->ins_h, s->window[s->strstart+1]);
#if MIN_MATCH != 3
            Call UPDATE_HASH() MIN_MATCH-3 more times
#endif
        }
        /* If the whole input has less than MIN_MATCH bytes, ins_h is garbage,
         * but this is not important since only literal bytes will be emitted.
         */

    } while (s->lookahead < MIN_LOOKAHEAD && s->strm->avail_in != 0);
}

/* ===========================================================================
 * Flush the current block, with given end-of-file flag.
 * IN assertion: strstart is set to the end of the current match.
 */
#define FLUSH_BLOCK_ONLY(s, eof) { \
   _tr_flush_block(s, (s->block_start >= 0L ? \
                   (charf *)&s->window[(unsigned)s->block_start] : \
                   (charf *)Z_NULL), \
                (ulg)((long)s->strstart - s->block_start), \
                (eof)); \
   s->block_start = s->strstart; \
   flush_pending(s->strm); \
   Tracev((stderr,"[FLUSH]")); \
}

/* Same but force premature exit if necessary. */
#define FLUSH_BLOCK(s, eof) { \
   FLUSH_BLOCK_ONLY(s, eof); \
   if (s->strm->avail_out == 0) return (eof) ? finish_started : need_more; \
}

/* ===========================================================================
 * Copy without compression as much as possible from the input stream, return
 * the current block state.
 * This function does not insert new strings in the dictionary since
 * uncompressible data is probably not useful. This function is used
 * only for the level=0 compression option.
 * NOTE: this function should be optimized to avoid extra copying from
 * window to pending_buf.
 */
local block_state deflate_stored(s, flush)
```

```
1052     deflate_state *s;
1053     int flush;
1054 {
1055     /* Stored blocks are limited to 0xffff bytes, pending_buf is limited
1056      * to pending_buf_size, and each stored block has a 5 byte header:
1057      */
1058     ulg max_block_size = 0xffff;
1059     ulg max_start;
1060 
1061     if (max_block_size > s->pending_buf_size - 5) {
1062         max_block_size = s->pending_buf_size - 5;
1063     }
1064 
1065     /* Copy as much as possible from input to output: */
1066     for (;;) {
1067         /* Fill the window as much as possible: */
1068         if (s->lookahead <= 1) {
1069 
1070             Assert(s->strstart < s->w_size+MAX_DIST(s) ||
1071                    s->block_start >= (long)s->w_size, "slide too late");
1072 
1073             fill_window(s);
1074             if (s->lookahead == 0 && flush == Z_NO_FLUSH) return need_more;
1075 
1076             if (s->lookahead == 0) break; /* flush the current block */
1077         }
1078         Assert(s->block_start >= 0L, "block gone");
1079 
1080         s->strstart += s->lookahead;
1081         s->lookahead = 0;
1082 
1083         /* Emit a stored block if pending_buf will be full: */
1084         max_start = s->block_start + max_block_size;
1085         if (s->strstart == 0 || (ulg)s->strstart >= max_start) {
1086             /* strstart == 0 is possible when wraparound on 16-bit machine */
1087             s->lookahead = (uInt)(s->strstart - max_start);
1088             s->strstart = (uInt)max_start;
1089             FLUSH_BLOCK(s, 0);
1090         }
1091         /* Flush if we may have to slide, otherwise block_start may become
1092          * negative and the data will be gone:
1093          */
1094         if (s->strstart - (uInt)s->block_start >= MAX_DIST(s)) {
1095             FLUSH_BLOCK(s, 0);
1096         }
1097     }
1098     FLUSH_BLOCK(s, flush == Z_FINISH);
1099     return flush == Z_FINISH ? finish_done : block_done;
1100 }
1101 
1102 /* ===========================================================================
1103  * Compress as much as possible from the input stream, return the current
1104  * block state.
1105  * This function does not perform lazy evaluation of matches and inserts
1106  * new strings in the dictionary only for unmatched strings or for short
1107  * matches. It is used only for the fast compression options.
1108  */
1109 local block_state deflate_fast(s, flush)
1110     deflate_state *s;
1111     int flush;
1112 {
1113     IPos hash_head = NIL; /* head of the hash chain */
1114     int bflush;           /* set if current block must be flushed */
1115 
1116     for (;;) {
1117         /* Make sure that we always have enough lookahead, except
```



```
1118          * at the end of the input file. We need MAX_MATCH bytes
1119          * for the next match, plus MIN_MATCH bytes to insert the
1120          * string following the next match.
1121          */
1122         if (s->lookahead < MIN_LOOKAHEAD) {
1123             fill_window(s);
1124             if (s->lookahead < MIN_LOOKAHEAD && flush == Z_NO_FLUSH) {
1125                 return need_more;
1126             }
1127             if (s->lookahead == 0) break; /* flush the current block */
1128         }
1129 
1130         /* Insert the string window[strstart .. strstart+2] in the
1131          * dictionary, and set hash_head to the head of the hash chain:
1132          */
1133         if (s->lookahead >= MIN_MATCH) {
1134             INSERT_STRING(s, s->strstart, hash_head);
1135         }
1136 
1137         /* Find the longest match, discarding those <= prev_length.
1138          * At this point we have always match_length < MIN_MATCH
1139          */
1140         if (hash_head != NIL && s->strstart - hash_head <= MAX_DIST(s)) {
1141             /* To simplify the code, we prevent matches with the string
1142              * of window index 0 (in particular we have to avoid a match
1143              * of the string with itself at the start of the input file).
1144              */
1145             if (s->strategy != Z_HUFFMAN_ONLY) {
1146                 s->match_length = longest_match (s, hash_head);
1147             }
1148             /* longest_match() sets match_start */
1149         }
1150         if (s->match_length >= MIN_MATCH) {
1151             check_match(s, s->strstart, s->match_start, s->match_length);
1152 
1153             _tr_tally_dist(s, s->strstart - s->match_start,
1154                            s->match_length - MIN_MATCH, bflush);
1155 
1156             s->lookahead -= s->match_length;
1157 
1158             /* Insert new strings in the hash table only if the match length
1159              * is not too large. This saves time but degrades compression.
1160              */
1161 #ifndef FASTEST
1162             if (s->match_length <= s->max_insert_length &&
1163                 s->lookahead >= MIN_MATCH) {
1164                 s->match_length--; /* string at strstart already in hash table */
1165                 do {
1166                     s->strstart++;
1167                     INSERT_STRING(s, s->strstart, hash_head);
1168                     /* strstart never exceeds WSIZE-MAX_MATCH, so there are
1169                      * always MIN_MATCH bytes ahead.
1170                      */
1171                 } while (--s->match_length != 0);
1172                 s->strstart++;
1173             } else
1174 #endif
1175             {
1176                 s->strstart += s->match_length;
1177                 s->match_length = 0;
1178                 s->ins_h = s->window[s->strstart];
1179                 UPDATE_HASH(s, s->ins_h, s->window[s->strstart+1]);
1180 #if MIN_MATCH != 3
1181                 Call UPDATE_HASH() MIN_MATCH-3 more times
1182 #endif
1183                 /* If lookahead < MIN_MATCH, ins_h is garbage, but it does not
```

```
1184                 * matter since it will be recomputed at next deflate call.
1185                 */
1186            }
1187        } else {
1188            /* No match, output a literal byte */
1189            Tracevv((stderr,"%c", s->window[s->strstart]));
1190            _tr_tally_lit (s, s->window[s->strstart], bflush);
1191            s->lookahead--;
1192            s->strstart++;
1193        }
1194        if (bflush) FLUSH_BLOCK(s, 0);
1195    }
1196    FLUSH_BLOCK(s, flush == Z_FINISH);
1197    return flush == Z_FINISH ? finish_done : block_done;
1198 }

1200 /* ===========================================================================
1201  * Same as above, but achieves better compression. We use a lazy
1202  * evaluation for matches: a match is finally adopted only if there is
1203  * no better match at the next window position.
1204  */
1205 local block_state deflate_slow(s, flush)
1206     deflate_state *s;
1207     int flush;
1208 {
1209     IPos hash_head = NIL;    /* head of hash chain */
1210     int bflush;              /* set if current block must be flushed */

1212     /* Process the input block. */
1213     for (;;) {
1214         /* Make sure that we always have enough lookahead, except
1215          * at the end of the input file. We need MAX_MATCH bytes
1216          * for the next match, plus MIN_MATCH bytes to insert the
1217          * string following the next match.
1218          */
1219         if (s->lookahead < MIN_LOOKAHEAD) {
1220             fill_window(s);
1221             if (s->lookahead < MIN_LOOKAHEAD && flush == Z_NO_FLUSH) {
1222                 return need_more;
1223             }
1224             if (s->lookahead == 0) break; /* flush the current block */
1225         }

1227         /* Insert the string window[strstart .. strstart+2] in the
1228          * dictionary, and set hash_head to the head of the hash chain:
1229          */
1230         if (s->lookahead >= MIN_MATCH) {
1231             INSERT_STRING(s, s->strstart, hash_head);
1232         }

1234         /* Find the longest match, discarding those <= prev_length.
1235          */
1236         s->prev_length = s->match_length, s->prev_match = s->match_start;
1237         s->match_length = MIN_MATCH-1;

1239         if (hash_head != NIL && s->prev_length < s->max_lazy_match &&
1240             s->strstart - hash_head <= MAX_DIST(s)) {
1241             /* To simplify the code, we prevent matches with the string
1242              * of window index 0 (in particular we have to avoid a match
1243              * of the string with itself at the start of the input file).
1244              */
1245             if (s->strategy != Z_HUFFMAN_ONLY) {
1246                 s->match_length = longest_match (s, hash_head);
1247             }
1248             /* longest_match() sets match_start */
```



```
1250             if (s->match_length <= 5 && (s->strategy == Z_FILTERED ||
1251                  (s->match_length == MIN_MATCH &&
1252                   s->strstart - s->match_start > TOO_FAR))) {

1254                 /* If prev_match is also MIN_MATCH, match_start is garbage
1255                  * but we will ignore the current match anyway.
1256                  */
1257                 s->match_length = MIN_MATCH-1;
1258             }
1259         }
1260         /* If there was a match at the previous step and the current
1261          * match is not better, output the previous match:
1262          */
1263         if (s->prev_length >= MIN_MATCH && s->match_length <= s->prev_length) {
1264             uInt max_insert = s->strstart + s->lookahead - MIN_MATCH;
1265             /* Do not insert strings in hash table beyond this. */

1267             check_match(s, s->strstart-1, s->prev_match, s->prev_length);

1269             _tr_tally_dist(s, s->strstart -1 - s->prev_match,
1270                            s->prev_length - MIN_MATCH, bflush);

1272             /* Insert in hash table all strings up to the end of the match.
1273              * strstart-1 and strstart are already inserted. If there is not
1274              * enough lookahead, the last two strings are not inserted in
1275              * the hash table.
1276              */
1277             s->lookahead -= s->prev_length-1;
1278             s->prev_length -= 2;
1279             do {
1280                 if (++s->strstart <= max_insert) {
1281                     INSERT_STRING(s, s->strstart, hash_head);
1282                 }
1283             } while (--s->prev_length != 0);
1284             s->match_available = 0;
1285             s->match_length = MIN_MATCH-1;
1286             s->strstart++;

1288             if (bflush) FLUSH_BLOCK(s, 0);

1290         } else if (s->match_available) {
1291             /* If there was no match at the previous position, output a
1292              * single literal. If there was a match but the current match
1293              * is longer, truncate the previous match to a single literal.
1294              */
1295             Tracevv((stderr,"%c", s->window[s->strstart-1]));
1296             _tr_tally_lit(s, s->window[s->strstart-1], bflush);
1297             if (bflush) {
1298                 FLUSH_BLOCK_ONLY(s, 0);
1299             }
1300             s->strstart++;
1301             s->lookahead--;
1302             if (s->strm->avail_out == 0) return need_more;
1303         } else {
1304             /* There is no previous match to compare with, wait for
1305              * the next step to decide.
1306              */
1307             s->match_available = 1;
1308             s->strstart++;
1309             s->lookahead--;
1310         }
1311     }
1312     Assert (flush != Z_NO_FLUSH, "no flush?");
1313     if (s->match_available) {
1314         Tracevv((stderr,"%c", s->window[s->strstart-1]));
1315         _tr_tally_lit(s, s->window[s->strstart-1], bflush);
```

```
1316         s->match_available = 0;
1317     }
1318     FLUSH_BLOCK(s, flush == Z_FINISH);
1319     return flush == Z_FINISH ? finish_done : block_done;
1320 }
```

```
**********************************************************
   11765 Wed Jan 31 21:21:18 2007
new/usr/src/uts/common/zmod/deflate.h
1000001 add zlib compression support
**********************************************************
```

```
/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "@(#)deflate.h  1.2     07/01/31 SMI"

/*
 * deflate.h -- internal compression state
 * Copyright (C) 1995-1998 Jean-loup Gailly
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

/* @(#) $Id$ */

#ifndef _DEFLATE_H
#define _DEFLATE_H

#include "zutil.h"

/* ===========================================================================
 * Internal compression state.
 */

#define LENGTH_CODES 29
/* number of length codes, not counting the special END_BLOCK code */

#define LITERALS  256
/* number of literal bytes 0..255 */

#define L_CODES (LITERALS+1+LENGTH_CODES)
/* number of Literal or Length codes, including the END_BLOCK code */

#define D_CODES   30
/* number of distance codes */

#define BL_CODES  19
/* number of codes used to transfer the bit lengths */

#define HEAP_SIZE (2*L_CODES+1)
/* maximum heap size */

#define MAX_BITS 15
/* All codes must not exceed MAX_BITS bits */

#define INIT_STATE    42
#define BUSY_STATE   113
#define FINISH_STATE 666
/* Stream status */


/* Data structure describing a single value and its code string. */
typedef struct ct_data_s {
    union {
        ush  freq;       /* frequency count */
        ush  code;       /* bit string */
```



```
    } fc;
    union {
        ush  dad;        /* father node in Huffman tree */
        ush  len;        /* length of bit string */
    } dl;
} FAR ct_data;

#define Freq fc.freq
#define Code fc.code
#define Dad  dl.dad
#define Len  dl.len

typedef struct static_tree_desc_s  static_tree_desc;

typedef struct tree_desc_s {
    ct_data *dyn_tree;           /* the dynamic tree */
    int     max_code;            /* largest code with non zero frequency */
    static_tree_desc *stat_desc; /* the corresponding static tree */
} FAR tree_desc;

typedef ush Pos;
typedef Pos FAR Posf;
typedef unsigned IPos;

/* A Pos is an index in the character window. We use short instead of int to
 * save space in the various tables. IPos is used only for parameter passing.
 */

typedef struct internal_state {
    z_streamp strm;      /* pointer back to this zlib stream */
    int   status;        /* as the name implies */
    Bytef *pending_buf;  /* output still pending */
    ulg   pending_buf_size; /* size of pending_buf */
    Bytef *pending_out;  /* next pending byte to output to the stream */
    int   pending;       /* nb of bytes in the pending buffer */
    int   noheader;      /* suppress zlib header and adler32 */
    Byte  data_type;     /* UNKNOWN, BINARY or ASCII */
    Byte  method;        /* STORED (for zip only) or DEFLATED */
    int   last_flush;    /* value of flush param for previous deflate call */

                /* used by deflate.c: */

    uInt  w_size;        /* LZ77 window size (32K by default) */
    uInt  w_bits;        /* log2(w_size)  (8..16) */
    uInt  w_mask;        /* w_size - 1 */

    Bytef *window;
    /* Sliding window. Input bytes are read into the second half of the window,
     * and move to the first half later to keep a dictionary of at least wSize
     * bytes. With this organization, matches are limited to a distance of
     * wSize-MAX_MATCH bytes, but this ensures that IO is always
     * performed with a length multiple of the block size. Also, it limits
     * the window size to 64K, which is quite useful on MSDOS.
     * To do: use the user input buffer as sliding window.
     */

    ulg window_size;
    /* Actual size of window: 2*wSize, except when the user input buffer
     * is directly used as sliding window.
     */

    Posf *prev;
    /* Link to older string with same hash index. To limit the size of this
     * array to 64K, this link is maintained only for the last 32K strings.
     * An index in this array is thus a window index modulo 32K.
     */
```

```
129     Posf *head; /* Heads of the hash chains or NIL. */

131     uInt  ins_h;          /* hash index of string to be inserted */
132     uInt  hash_size;      /* number of elements in hash table */
133     uInt  hash_bits;      /* log2(hash_size) */
134     uInt  hash_mask;      /* hash_size-1 */

136     uInt  hash_shift;
137     /* Number of bits by which ins_h must be shifted at each input
138      * step. It must be such that after MIN_MATCH steps, the oldest
139      * byte no longer takes part in the hash key, that is:
140      *   hash_shift * MIN_MATCH >= hash_bits
141      */

143     long block_start;
144     /* Window position at the beginning of the current output block. Gets
145      * negative when the window is moved backwards.
146      */

148     uInt match_length;           /* length of best match */
149     IPos prev_match;             /* previous match */
150     int match_available;         /* set if previous match exists */
151     uInt strstart;               /* start of string to insert */
152     uInt match_start;            /* start of matching string */
153     uInt lookahead;              /* number of valid bytes ahead in window */

155     uInt prev_length;
156     /* Length of the best match at previous step. Matches not greater than this
157      * are discarded. This is used in the lazy match evaluation.
158      */

160     uInt max_chain_length;
161     /* To speed up deflation, hash chains are never searched beyond this
162      * length.  A higher limit improves compression ratio but degrades the
163      * speed.
164      */

166     uInt max_lazy_match;
167     /* Attempt to find a better match only when the current match is strictly
168      * smaller than this value. This mechanism is used only for compression
169      * levels >= 4.
170      */
171 #   define max_insert_length  max_lazy_match
172     /* Insert new strings in the hash table only if the match length is not
173      * greater than this length. This saves time but degrades compression.
174      * max_insert_length is used only for compression levels <= 3.
175      */

177     int level;    /* compression level (1..9) */
178     int strategy; /* favor or force Huffman coding*/

180     uInt good_match;
181     /* Use a faster search when the previous match is longer than this */

183     int nice_match; /* Stop searching when current match exceeds this */

185                 /* used by trees.c: */
186     /* Didn't use ct_data typedef below to supress compiler warning */
187     struct ct_data_s dyn_ltree[HEAP_SIZE];   /* literal and length tree */
188     struct ct_data_s dyn_dtree[2*D_CODES+1]; /* distance tree */
189     struct ct_data_s bl_tree[2*BL_CODES+1];  /* Huffman tree for bit lengths */

191     struct tree_desc_s l_desc;               /* desc. for literal tree */
192     struct tree_desc_s d_desc;               /* desc. for distance tree */
193     struct tree_desc_s bl_desc;              /* desc. for bit length tree */
```



```
195     ush bl_count[MAX_BITS+1];
196     /* number of codes at each bit length for an optimal tree */

198     int heap[2*L_CODES+1];      /* heap used to build the Huffman trees */
199     int heap_len;               /* number of elements in the heap */
200     int heap_max;               /* element of largest frequency */
201     /* The sons of heap[n] are heap[2*n] and heap[2*n+1]. heap[0] is not used.
202      * The same heap array is used to build all trees.
203      */

205     uch depth[2*L_CODES+1];
206     /* Depth of each subtree used as tie breaker for trees of equal frequency
207      */

209     uchf *l_buf;          /* buffer for literals or lengths */

211     uInt  lit_bufsize;
212     /* Size of match buffer for literals/lengths.  There are 4 reasons for
213      * limiting lit_bufsize to 64K:
214      *   - frequencies can be kept in 16 bit counters
215      *   - if compression is not successful for the first block, all input
216      *     data is still in the window so we can still emit a stored block even
217      *     when input comes from standard input.  (This can also be done for
218      *     all blocks if lit_bufsize is not greater than 32K.)
219      *   - if compression is not successful for a file smaller than 64K, we can
220      *     even emit a stored file instead of a stored block (saving 5 bytes).
221      *     This is applicable only for zip (not gzip or zlib).
222      *   - creating new Huffman trees less frequently may not provide fast
223      *     adaptation to changes in the input data statistics. (Take for
224      *     example a binary file with poorly compressible code followed by
225      *     a highly compressible string table.) Smaller buffer sizes give
226      *     fast adaptation but have of course the overhead of transmitting
227      *     trees more frequently.
228      *   - I can't count above 4
229      */

231     uInt last_lit;      /* running index in l_buf */

233     ushf *d_buf;
234     /* Buffer for distances. To simplify the code, d_buf and l_buf have
235      * the same number of elements. To use different lengths, an extra flag
236      * array would be necessary.
237      */

239     ulg opt_len;        /* bit length of current block with optimal trees */
240     ulg static_len;     /* bit length of current block with static trees */
241     uInt matches;       /* number of string matches in current block */
242     int last_eob_len;   /* bit length of EOB code for last block */

244 #ifdef DEBUG
245     ulg compressed_len; /* total bit length of compressed file mod 2^32 */
246     ulg bits_sent;      /* bit length of compressed data sent mod 2^32 */
247 #endif

249     ush bi_buf;
250     /* Output buffer. bits are inserted starting at the bottom (least
251      * significant bits).
252      */
253     int bi_valid;
254     /* Number of valid bits in bi_buf.  All bits above the last valid bit
255      * are always zero.
256      */

258 } FAR deflate_state;
```

```
/* Output a byte on the stream.
 * IN assertion: there is enough room in pending_buf.
 */
#define put_byte(s, c) {s->pending_buf[s->pending++] = (c);}


#define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)
/* Minimum amount of lookahead, except at the end of the input file.
 * See deflate.c for comments about the MIN_MATCH+1.
 */

#define MAX_DIST(s)  ((s)->w_size-MIN_LOOKAHEAD)
/* In order to simplify the code, particularly on 16 bit machines, match
 * distances are limited to MAX_DIST instead of WSIZE.
 */

        /* in trees.c */
void _tr_init         OF((deflate_state *s));
int  _tr_tally        OF((deflate_state *s, unsigned dist, unsigned lc));
void _tr_flush_block  OF((deflate_state *s, charf *buf, ulg stored_len,
                          int eof));
void _tr_align        OF((deflate_state *s));
void _tr_stored_block OF((deflate_state *s, charf *buf, ulg stored_len,
                          int eof));

#define d_code(dist) \
   ((dist) < 256 ? _dist_code[dist] : _dist_code[256+((dist)>>7)])
/* Mapping from a distance to a distance code. dist is the distance - 1 and
 * must not have side effects. _dist_code[256] and _dist_code[257] are never
 * used.
 */

#ifndef DEBUG
/* Inline versions of _tr_tally for speed: */

#if defined(GEN_TREES_H) || !defined(STDC)
  extern uch _length_code[];
  extern uch _dist_code[];
#else
  extern const uch _length_code[];
  extern const uch _dist_code[];
#endif

# define _tr_tally_lit(s, c, flush) \
  { uch cc = (c); \
    s->d_buf[s->last_lit] = 0; \
    s->l_buf[s->last_lit++] = cc; \
    s->dyn_ltree[cc].Freq++; \
    flush = (s->last_lit == s->lit_bufsize-1); \
   }
# define _tr_tally_dist(s, distance, length, flush) \
  { uch len = (length); \
    ush dist = (distance); \
    s->d_buf[s->last_lit] = dist; \
    s->l_buf[s->last_lit++] = len; \
    dist--; \
    s->dyn_ltree[_length_code[len]+LITERALS+1].Freq++; \
    s->dyn_dtree[d_code(dist)].Freq++; \
    flush = (s->last_lit == s->lit_bufsize-1); \
  }
#else
# define _tr_tally_lit(s, c, flush) flush = _tr_tally(s, 0, c)
# define _tr_tally_dist(s, distance, length, flush) \
              flush = _tr_tally(s, distance, length)
#endif
```



```
#endif
```

```
**********************************************************
   42694 Wed Jan 31 21:21:18 2007
new/usr/src/uts/common/zmod/trees.c
1000001 add zlib compression support
**********************************************************
```

```
/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "@(#)trees.c    1.2     07/01/31 SMI"

/*
 * trees.c -- output deflated data using Huffman coding
 * Copyright (C) 1995-1998 Jean-loup Gailly
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 *  ALGORITHM
 *
 *      The "deflation" process uses several Huffman trees. The more
 *      common source values are represented by shorter bit sequences.
 *
 *      Each code tree is stored in a compressed form which is itself
 * a Huffman encoding of the lengths of all the code strings (in
 * ascending order by source values).  The actual code strings are
 * reconstructed from the lengths in the inflate process, as described
 * in the deflate specification.
 *
 *  REFERENCES
 *
 *      Deutsch, L.P.,"'Deflate' Compressed Data Format Specification".
 *      Available in ftp.uu.net:/pub/archiving/zip/doc/deflate-1.1.doc
 *
 *      Storer, James A.
 *          Data Compression:  Methods and Theory, pp. 49-50.
 *          Computer Science Press, 1988.  ISBN 0-7167-8156-5.
 *
 *      Sedgewick, R.
 *          Algorithms, p290.
 *          Addison-Wesley, 1983. ISBN 0-201-06672-6.
 */

/* @(#) $Id$ */

/* #define GEN_TREES_H */

#include "deflate.h"

/* ===========================================================================
 * Constants
 */

#define MAX_BL_BITS 7
/* Bit length codes must not exceed MAX_BL_BITS bits */

#define END_BLOCK 256
/* end of block literal code */

#define REP_3_6      16
/* repeat previous bit length 3-6 times (2 bits of repeat count) */

#define REPZ_3_10    17
/* repeat a zero length 3-10 times  (3 bits of repeat count) */
```



```
#define REPZ_11_138  18
/* repeat a zero length 11-138 times  (7 bits of repeat count) */

local const int extra_lbits[LENGTH_CODES] /* extra bits for each length code */
   = {0,0,0,0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,0};

local const int extra_dbits[D_CODES] /* extra bits for each distance code */
   = {0,0,0,0,1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,10,10,11,11,12,12,13,13};

local const int extra_blbits[BL_CODES]/* extra bits for each bit length code */
   = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,3,7};

local const uch bl_order[BL_CODES]
   = {16,17,18,0,8,7,9,6,10,5,11,4,12,3,13,2,14,1,15};
/* The lengths of the bit length codes are sent in order of decreasing
 * probability, to avoid transmitting the lengths for unused bit length codes.
 */

#define Buf_size (8 * 2*sizeof(char))
/* Number of bits used within bi_buf. (bi_buf might be implemented on
 * more than 16 bits on some systems.)
 */

/* ===========================================================================
 * Local data. These are initialized only once.
 */

#define DIST_CODE_LEN  512 /* see definition of array dist_code below */

#if defined(GEN_TREES_H) || !defined(STDC)
/* non ANSI compilers may not accept trees.h */

local ct_data static_ltree[L_CODES+2];
/* The static literal tree. Since the bit lengths are imposed, there is no
 * need for the L_CODES extra codes used during heap construction. However
 * The codes 286 and 287 are needed to build a canonical tree (see _tr_init
 * below).
 */

local ct_data static_dtree[D_CODES];
/* The static distance tree. (Actually a trivial tree since all codes use
 * 5 bits.)
 */

uch _dist_code[DIST_CODE_LEN];
/* Distance codes. The first 256 values correspond to the distances
 * 3 .. 258, the last 256 values correspond to the top 8 bits of
 * the 15 bit distances.
 */

uch _length_code[MAX_MATCH-MIN_MATCH+1];
/* length code for each normalized match length (0 == MIN_MATCH) */

local int base_length[LENGTH_CODES];
/* First normalized length for each code (0 = MIN_MATCH) */

local int base_dist[D_CODES];
/* First normalized distance for each code (0 = distance of 1) */

#else
#  include "trees.h"
#endif /* GEN_TREES_H */

struct static_tree_desc_s {
    const ct_data *static_tree;  /* static tree or NULL */
    const intf *extra_bits;      /* extra bits for each code or NULL */
```

```
128     int     extra_base;          /* base index for extra_bits */
129     int     elems;               /* max number of elements in the tree */
130     int     max_length;          /* max bit length for the codes */
131 };

133 local static_tree_desc  static_l_desc =
134 {static_ltree, extra_lbits, LITERALS+1, L_CODES, MAX_BITS};

136 local static_tree_desc  static_d_desc =
137 {static_dtree, extra_dbits, 0,          D_CODES, MAX_BITS};

139 local static_tree_desc  static_bl_desc =
140 {(const ct_data *)0, extra_blbits, 0,   BL_CODES, MAX_BL_BITS};

142 /* ===========================================================================
143  * Local (static) routines in this file.
144  */

146 local void tr_static_init OF((void));
147 local void init_block     OF((deflate_state *s));
148 local void pqdownheap     OF((deflate_state *s, ct_data *tree, int k));
149 local void gen_bitlen     OF((deflate_state *s, tree_desc *desc));
150 local void gen_codes      OF((ct_data *tree, int max_code, ushf *bl_count));
151 local void build_tree     OF((deflate_state *s, tree_desc *desc));
152 local void scan_tree      OF((deflate_state *s, ct_data *tree, int max_code));
153 local void send_tree      OF((deflate_state *s, ct_data *tree, int max_code));
154 local int  build_bl_tree  OF((deflate_state *s));
155 local void send_all_trees OF((deflate_state *s, int lcodes, int dcodes,
156                               int blcodes));
157 local void compress_block OF((deflate_state *s, ct_data *ltree,
158                               ct_data *dtree));
159 local void set_data_type  OF((deflate_state *s));
160 local unsigned bi_reverse OF((unsigned value, int length));
161 local void bi_windup      OF((deflate_state *s));
162 local void bi_flush       OF((deflate_state *s));
163 local void copy_block     OF((deflate_state *s, charf *buf, unsigned len,
164                               int header));

166 #ifdef GEN_TREES_H
167 local void gen_trees_header OF((void));
168 #endif

170 #  define send_code(s, c, tree) send_bits(s, tree[c].Code, tree[c].Len)
171    /* Send a code of the given tree. c and tree must not have side effects */

173 /* ===========================================================================
174  * Output a short LSB first on the stream.
175  * IN assertion: there is enough room in pendingBuf.
176  */
177 #define put_short(s, w) { \
178     put_byte(s, (uch)((w) & 0xff)); \
179     put_byte(s, (uch)((ush)(w) >> 8)); \
180 }

182 /* ===========================================================================
183  * Send a value on a given number of bits.
184  * IN assertion: length <= 16 and value fits in length bits.
185  */

187 #define send_bits(s, value, length) \
188 { int len = length;\
189   if (s->bi_valid > (int)Buf_size - len) {\
190     int val = value;\
191     s->bi_buf |= (val << s->bi_valid);\
192     put_short(s, s->bi_buf);\
193     s->bi_buf = (ush)val >> (Buf_size - s->bi_valid);\
```



```
194     s->bi_valid += len - Buf_size;\
195   } else {\
196     s->bi_buf |= (value) << s->bi_valid;\
197     s->bi_valid += len;\
198   }\
199 }


202 #define MAX(a,b) (a >= b ? a : b)
203 /* the arguments must not have side effects */

205 /* ===========================================================================
206  * Initialize the various 'constant' tables.
207  */
208 local void tr_static_init()
209 {
210 #if defined(GEN_TREES_H) || !defined(STDC)
211     static int static_init_done = 0;
212     int n;        /* iterates over tree elements */
213     int bits;     /* bit counter */
214     int length;   /* length value */
215     int code;     /* code value */
216     int dist;     /* distance index */
217     ush bl_count[MAX_BITS+1];
218     /* number of codes at each bit length for an optimal tree */

220     if (static_init_done) return;

222     /* For some embedded targets, global variables are not initialized: */
223     static_l_desc.static_tree = static_ltree;
224     static_l_desc.extra_bits = extra_lbits;
225     static_d_desc.static_tree = static_dtree;
226     static_d_desc.extra_bits = extra_dbits;
227     static_bl_desc.extra_bits = extra_blbits;

229     /* Initialize the mapping length (0..255) -> length code (0..28) */
230     length = 0;
231     for (code = 0; code < LENGTH_CODES-1; code++) {
232         base_length[code] = length;
233         for (n = 0; n < (1<<extra_lbits[code]); n++) {
234             _length_code[length++] = (uch)code;
235         }
236     }
237     Assert (length == 256, "tr_static_init: length != 256");
238     /* Note that the length 255 (match length 258) can be represented
239      * in two different ways: code 284 + 5 bits or code 285, so we
240      * overwrite length_code[255] to use the best encoding:
241      */
242     _length_code[length-1] = (uch)code;

244     /* Initialize the mapping dist (0..32K) -> dist code (0..29) */
245     dist = 0;
246     for (code = 0 ; code < 16; code++) {
247         base_dist[code] = dist;
248         for (n = 0; n < (1<<extra_dbits[code]); n++) {
249             _dist_code[dist++] = (uch)code;
250         }
251     }
252     Assert (dist == 256, "tr_static_init: dist != 256");
253     dist >>= 7; /* from now on, all distances are divided by 128 */
254     for ( ; code < D_CODES; code++) {
255         base_dist[code] = dist << 7;
256         for (n = 0; n < (1<<(extra_dbits[code]-7)); n++) {
257             _dist_code[256 + dist++] = (uch)code;
258         }
259     }
```

```
260     Assert (dist == 256, "tr_static_init: 256+dist != 512");

262     /* Construct the codes of the static literal tree */
263     for (bits = 0; bits <= MAX_BITS; bits++) bl_count[bits] = 0;
264     n = 0;
265     while (n <= 143) static_ltree[n++].Len = 8, bl_count[8]++;
266     while (n <= 255) static_ltree[n++].Len = 9, bl_count[9]++;
267     while (n <= 279) static_ltree[n++].Len = 7, bl_count[7]++;
268     while (n <= 287) static_ltree[n++].Len = 8, bl_count[8]++;
269     /* Codes 286 and 287 do not exist, but we must include them in the
270      * tree construction to get a canonical Huffman tree (longest code
271      * all ones)
272      */
273     gen_codes((ct_data *)static_ltree, L_CODES+1, bl_count);

275     /* The static distance tree is trivial: */
276     for (n = 0; n < D_CODES; n++) {
277         static_dtree[n].Len = 5;
278         static_dtree[n].Code = bi_reverse((unsigned)n, 5);
279     }
280     static_init_done = 1;

282 #  ifdef GEN_TREES_H
283     gen_trees_header();
284 #  endif
285 #endif /* defined(GEN_TREES_H) || !defined(STDC) */
286 }

288 /* ===========================================================================
289  * Genererate the file trees.h describing the static trees.
290  */
291 #ifdef GEN_TREES_H
292 #  ifndef DEBUG
293 #    include <stdio.h>
294 #  endif

296 #  define SEPARATOR(i, last, width) \
297       ((i) == (last)? "\n};\n\n" :    \
298        ((i) % (width) == (width)-1 ? ",\n" : ", "))

300 void gen_trees_header()
301 {
302     FILE *header = fopen("trees.h", "w");
303     int i;

305     Assert (header != NULL, "Can't open trees.h");
306     fprintf(header,
307             "/* header created automatically with -DGEN_TREES_H */\n\n");

309     fprintf(header, "local const ct_data static_ltree[L_CODES+2] = {\n");
310     for (i = 0; i < L_CODES+2; i++) {
311         fprintf(header, "{{%3u},{%3u}}%s", static_ltree[i].Code,
312                 static_ltree[i].Len, SEPARATOR(i, L_CODES+1, 5));
313     }

315     fprintf(header, "local const ct_data static_dtree[D_CODES] = {\n");
316     for (i = 0; i < D_CODES; i++) {
317         fprintf(header, "{{%2u},{%2u}}%s", static_dtree[i].Code,
318                 static_dtree[i].Len, SEPARATOR(i, D_CODES-1, 5));
319     }

321     fprintf(header, "const uch _dist_code[DIST_CODE_LEN] = {\n");
322     for (i = 0; i < DIST_CODE_LEN; i++) {
323         fprintf(header, "%2u%s", _dist_code[i],
324                 SEPARATOR(i, DIST_CODE_LEN-1, 20));
325     }
```



```
327     fprintf(header, "const uch _length_code[MAX_MATCH-MIN_MATCH+1]= {\n");
328     for (i = 0; i < MAX_MATCH-MIN_MATCH+1; i++) {
329         fprintf(header, "%2u%s", _length_code[i],
330                 SEPARATOR(i, MAX_MATCH-MIN_MATCH, 20));
331     }

333     fprintf(header, "local const int base_length[LENGTH_CODES] = {\n");
334     for (i = 0; i < LENGTH_CODES; i++) {
335         fprintf(header, "%1u%s", base_length[i],
336                 SEPARATOR(i, LENGTH_CODES-1, 20));
337     }

339     fprintf(header, "local const int base_dist[D_CODES] = {\n");
340     for (i = 0; i < D_CODES; i++) {
341         fprintf(header, "%5u%s", base_dist[i],
342                 SEPARATOR(i, D_CODES-1, 10));
343     }

345     fclose(header);
346 }
347 #endif /* GEN_TREES_H */

349 /* ===========================================================================
350  * Initialize the tree data structures for a new zlib stream.
351  */
352 void _tr_init(s)
353     deflate_state *s;
354 {
355     tr_static_init();

357     s->l_desc.dyn_tree = s->dyn_ltree;
358     s->l_desc.stat_desc = &static_l_desc;

360     s->d_desc.dyn_tree = s->dyn_dtree;
361     s->d_desc.stat_desc = &static_d_desc;

363     s->bl_desc.dyn_tree = s->bl_tree;
364     s->bl_desc.stat_desc = &static_bl_desc;

366     s->bi_buf = 0;
367     s->bi_valid = 0;
368     s->last_eob_len = 8; /* enough lookahead for inflate */
369 #ifdef DEBUG
370     s->compressed_len = 0L;
371     s->bits_sent = 0L;
372 #endif

374     /* Initialize the first block of the first file: */
375     init_block(s);
376 }

378 /* ===========================================================================
379  * Initialize a new block.
380  */
381 local void init_block(s)
382     deflate_state *s;
383 {
384     int n; /* iterates over tree elements */

386     /* Initialize the trees. */
387     for (n = 0; n < L_CODES;  n++) s->dyn_ltree[n].Freq = 0;
388     for (n = 0; n < D_CODES;  n++) s->dyn_dtree[n].Freq = 0;
389     for (n = 0; n < BL_CODES; n++) s->bl_tree[n].Freq = 0;

391     s->dyn_ltree[END_BLOCK].Freq = 1;
```

```
392     s->opt_len = s->static_len = 0L;
393     s->last_lit = s->matches = 0;
394 }

396 #define SMALLEST 1
397 /* Index within the heap array of least frequent node in the Huffman tree */


400 /* ===========================================================================
401  * Remove the smallest element from the heap and recreate the heap with
402  * one less element. Updates heap and heap_len.
403  */
404 #define pqremove(s, tree, top) \
405 {\
406     top = s->heap[SMALLEST]; \
407     s->heap[SMALLEST] = s->heap[s->heap_len--]; \
408     pqdownheap(s, tree, SMALLEST); \
409 }

411 /* ===========================================================================
412  * Compares to subtrees, using the tree depth as tie breaker when
413  * the subtrees have equal frequency. This minimizes the worst case length.
414  */
415 #define smaller(tree, n, m, depth) \
416    (tree[n].Freq < tree[m].Freq || \
417    (tree[n].Freq == tree[m].Freq && depth[n] <= depth[m]))

419 /* ===========================================================================
420  * Restore the heap property by moving down the tree starting at node k,
421  * exchanging a node with the smallest of its two sons if necessary, stopping
422  * when the heap property is re-established (each father smaller than its
423  * two sons).
424  */
425 local void pqdownheap(s, tree, k)
426     deflate_state *s;
427     ct_data *tree;  /* the tree to restore */
428     int k;               /* node to move down */
429 {
430     int v = s->heap[k];
431     int j = k << 1;  /* left son of k */
432     while (j <= s->heap_len) {
433         /* Set j to the smallest of the two sons: */
434         if (j < s->heap_len &&
435             smaller(tree, s->heap[j+1], s->heap[j], s->depth)) {
436             j++;
437         }
438         /* Exit if v is smaller than both sons */
439         if (smaller(tree, v, s->heap[j], s->depth)) break;

441         /* Exchange v with the smallest son */
442         s->heap[k] = s->heap[j];  k = j;

444         /* And continue down the tree, setting j to the left son of k */
445         j <<= 1;
446     }
447     s->heap[k] = v;
448 }

450 /* ===========================================================================
451  * Compute the optimal bit lengths for a tree and update the total bit length
452  * for the current block.
453  * IN assertion: the fields freq and dad are set, heap[heap_max] and
454  *    above are the tree nodes sorted by increasing frequency.
455  * OUT assertions: the field len is set to the optimal bit length, the
456  *     array bl_count contains the frequencies for each bit length.
457  *     The length opt_len is updated; static_len is also updated if stree is
```



```
458  *     not null.
459  */
460 local void gen_bitlen(s, desc)
461     deflate_state *s;
462     tree_desc *desc;    /* the tree descriptor */
463 {
464     ct_data *tree        = desc->dyn_tree;
465     int max_code         = desc->max_code;
466     const ct_data *stree = desc->stat_desc->static_tree;
467     const intf *extra    = desc->stat_desc->extra_bits;
468     int base             = desc->stat_desc->extra_base;
469     int max_length       = desc->stat_desc->max_length;
470     int h;              /* heap index */
471     int n, m;           /* iterate over the tree elements */
472     int bits;           /* bit length */
473     int xbits;          /* extra bits */
474     ush f;              /* frequency */
475     int overflow = 0;   /* number of elements with bit length too large */

477     for (bits = 0; bits <= MAX_BITS; bits++) s->bl_count[bits] = 0;

479     /* In a first pass, compute the optimal bit lengths (which may
480      * overflow in the case of the bit length tree).
481      */
482     tree[s->heap[s->heap_max]].Len = 0; /* root of the heap */

484     for (h = s->heap_max+1; h < HEAP_SIZE; h++) {
485         n = s->heap[h];
486         bits = tree[tree[n].Dad].Len + 1;
487         if (bits > max_length) bits = max_length, overflow++;
488         tree[n].Len = (ush)bits;
489         /* We overwrite tree[n].Dad which is no longer needed */

491         if (n > max_code) continue; /* not a leaf node */

493         s->bl_count[bits]++;
494         xbits = 0;
495         if (n >= base) xbits = extra[n-base];
496         f = tree[n].Freq;
497         s->opt_len += (ulg)f * (bits + xbits);
498         if (stree) s->static_len += (ulg)f * (stree[n].Len + xbits);
499     }
500     if (overflow == 0) return;

502     Trace((stderr,"\nbit length overflow\n"));
503     /* This happens for example on obj2 and pic of the Calgary corpus */

505     /* Find the first bit length which could increase: */
506     do {
507         bits = max_length-1;
508         while (s->bl_count[bits] == 0) bits--;
509         s->bl_count[bits]--;      /* move one leaf down the tree */
510         s->bl_count[bits+1] += 2; /* move one overflow item as its brother */
511         s->bl_count[max_length]--;
512         /* The brother of the overflow item also moves one step up,
513          * but this does not affect bl_count[max_length]
514          */
515         overflow -= 2;
516     } while (overflow > 0);

518     /* Now recompute all bit lengths, scanning in increasing frequency.
519      * h is still equal to HEAP_SIZE. (It is simpler to reconstruct all
520      * lengths instead of fixing only the wrong ones. This idea is taken
521      * from 'ar' written by Haruhiko Okumura.)
522      */
523     for (bits = max_length; bits != 0; bits--) {
```

```
524         n = s->bl_count[bits];
525         while (n != 0) {
526             m = s->heap[--h];
527             if (m > max_code) continue;
528             if (tree[m].Len != (unsigned) bits) {
529                 Trace((stderr,"code %d bits %d->%d\n", m, tree[m].Len, bits));
530                 s->opt_len += ((long)bits - (long)tree[m].Len)
531                               *(long)tree[m].Freq;
532                 tree[m].Len = (ush)bits;
533             }
534             n--;
535         }
536     }
537 }

539 /* ========================================================================
540  * Generate the codes for a given tree and bit counts (which need not be
541  * optimal).
542  * IN assertion: the array bl_count contains the bit length statistics for
543  * the given tree and the field len is set for all tree elements.
544  * OUT assertion: the field code is set for all tree elements of non
545  *     zero code length.
546  */
547 local void gen_codes (tree, max_code, bl_count)
548     ct_data *tree;             /* the tree to decorate */
549     int max_code;              /* largest code with non zero frequency */
550     ushf *bl_count;            /* number of codes at each bit length */
551 {
552     ush next_code[MAX_BITS+1]; /* next code value for each bit length */
553     ush code = 0;              /* running code value */
554     int bits;                  /* bit index */
555     int n;                     /* code index */

557     /* The distribution counts are first used to generate the code values
558      * without bit reversal.
559      */
560     for (bits = 1; bits <= MAX_BITS; bits++) {
561         next_code[bits] = code = (code + bl_count[bits-1]) << 1;
562     }
563     /* Check that the bit counts in bl_count are consistent. The last code
564      * must be all ones.
565      */
566     Assert (code + bl_count[MAX_BITS]-1 == (1<<MAX_BITS)-1,
567             "inconsistent bit counts");
568     Tracev((stderr,"\ngen_codes: max_code %d ", max_code));

570     for (n = 0;  n <= max_code; n++) {
571         int len = tree[n].Len;
572         if (len == 0) continue;
573         /* Now reverse the bits */
574         tree[n].Code = bi_reverse(next_code[len]++, len);

576         Tracecv(tree != static_ltree, (stderr,"\nn %3d %c l %2d c %4x (%x) ",
577              n, (isgraph(n) ? n : ' '), len, tree[n].Code, next_code[len]-1));
578     }
579 }

581 /* ========================================================================
582  * Construct one Huffman tree and assigns the code bit strings and lengths.
583  * Update the total bit length for the current block.
584  * IN assertion: the field freq is set for all tree elements.
585  * OUT assertions: the fields len and code are set to the optimal bit length
586  *     and corresponding code. The length opt_len is updated; static_len is
587  *     also updated if stree is not null. The field max_code is set.
588  */
589 local void build_tree(s, desc)
```



```
590     deflate_state *s;
591     tree_desc *desc; /* the tree descriptor */
592 {
593     ct_data *tree         = desc->dyn_tree;
594     const ct_data *stree  = desc->stat_desc->static_tree;
595     int elems             = desc->stat_desc->elems;
596     int n, m;          /* iterate over heap elements */
597     int max_code = -1; /* largest code with non zero frequency */
598     int node;          /* new node being created */

600     /* Construct the initial heap, with least frequent element in
601      * heap[SMALLEST]. The sons of heap[n] are heap[2*n] and heap[2*n+1].
602      * heap[0] is not used.
603      */
604     s->heap_len = 0, s->heap_max = HEAP_SIZE;

606     for (n = 0; n < elems; n++) {
607         if (tree[n].Freq != 0) {
608             s->heap[++(s->heap_len)] = max_code = n;
609             s->depth[n] = 0;
610         } else {
611             tree[n].Len = 0;
612         }
613     }

615     /* The pkzip format requires that at least one distance code exists,
616      * and that at least one bit should be sent even if there is only one
617      * possible code. So to avoid special checks later on we force at least
618      * two codes of non zero frequency.
619      */
620     while (s->heap_len < 2) {
621         node = s->heap[++(s->heap_len)] = (max_code < 2 ? ++max_code : 0);
622         tree[node].Freq = 1;
623         s->depth[node] = 0;
624         s->opt_len--; if (stree) s->static_len -= stree[node].Len;
625         /* node is 0 or 1 so it does not have extra bits */
626     }
627     desc->max_code = max_code;

629     /* The elements heap[heap_len/2+1 .. heap_len] are leaves of the tree,
630      * establish sub-heaps of increasing lengths:
631      */
632     for (n = s->heap_len/2; n >= 1; n--) pqdownheap(s, tree, n);

634     /* Construct the Huffman tree by repeatedly combining the least two
635      * frequent nodes.
636      */
637     node = elems;              /* next internal node of the tree */
638     do {
639         pqremove(s, tree, n);  /* n = node of least frequency */
640         m = s->heap[SMALLEST]; /* m = node of next least frequency */

642         s->heap[--(s->heap_max)] = n; /* keep the nodes sorted by frequency */
643         s->heap[--(s->heap_max)] = m;

645         /* Create a new node father of n and m */
646         tree[node].Freq = tree[n].Freq + tree[m].Freq;
647         s->depth[node] = (uch) (MAX(s->depth[n], s->depth[m]) + 1);
648         tree[n].Dad = tree[m].Dad = (ush)node;
649 #ifdef DUMP_BL_TREE
650         if (tree == s->bl_tree) {
651             fprintf(stderr,"\nnode %d(%d), sons %d(%d) %d(%d)",
652                     node, tree[node].Freq, n, tree[n].Freq, m, tree[m].Freq);
653         }
654 #endif
655         /* and insert the new node in the heap */
```

```
656         s->heap[SMALLEST] = node++;
657         pqdownheap(s, tree, SMALLEST);
659     } while (s->heap_len >= 2);
661     s->heap[--(s->heap_max)] = s->heap[SMALLEST];
663     /* At this point, the fields freq and dad are set. We can now
664      * generate the bit lengths.
665      */
666     gen_bitlen(s, (tree_desc *)desc);
668     /* The field len is now set, we can generate the bit codes */
669     gen_codes ((ct_data *)tree, max_code, s->bl_count);
670 }
672 /* ===========================================================================
673  * Scan a literal or distance tree to determine the frequencies of the codes
674  * in the bit length tree.
675  */
676 local void scan_tree (s, tree, max_code)
677     deflate_state *s;
678     ct_data *tree;   /* the tree to be scanned */
679     int max_code;    /* and its largest code of non zero frequency */
680 {
681     int n;                     /* iterates over all tree elements */
682     int prevlen = -1;          /* last emitted length */
683     int curlen;                /* length of current code */
684     int nextlen = tree[0].Len; /* length of next code */
685     int count = 0;             /* repeat count of the current code */
686     int max_count = 7;         /* max repeat count */
687     int min_count = 4;         /* min repeat count */
689     if (nextlen == 0) max_count = 138, min_count = 3;
690     tree[max_code+1].Len = (ush)0xffff; /* guard */
692     for (n = 0; n <= max_code; n++) {
693         curlen = nextlen; nextlen = tree[n+1].Len;
694         if (++count < max_count && curlen == nextlen) {
695             continue;
696         } else if (count < min_count) {
697             s->bl_tree[curlen].Freq += count;
698         } else if (curlen != 0) {
699             if (curlen != prevlen) s->bl_tree[curlen].Freq++;
700             s->bl_tree[REP_3_6].Freq++;
701         } else if (count <= 10) {
702             s->bl_tree[REPZ_3_10].Freq++;
703         } else {
704             s->bl_tree[REPZ_11_138].Freq++;
705         }
706         count = 0; prevlen = curlen;
707         if (nextlen == 0) {
708             max_count = 138, min_count = 3;
709         } else if (curlen == nextlen) {
710             max_count = 6, min_count = 3;
711         } else {
712             max_count = 7, min_count = 4;
713         }
714     }
715 }
717 /* ===========================================================================
718  * Send a literal or distance tree in compressed form, using the codes in
719  * bl_tree.
720  */
721 local void send_tree (s, tree, max_code)
```



```
722     deflate_state *s;
723     ct_data *tree; /* the tree to be scanned */
724     int max_code;       /* and its largest code of non zero frequency */
725 {
726     int n;                     /* iterates over all tree elements */
727     int prevlen = -1;          /* last emitted length */
728     int curlen;                /* length of current code */
729     int nextlen = tree[0].Len; /* length of next code */
730     int count = 0;             /* repeat count of the current code */
731     int max_count = 7;         /* max repeat count */
732     int min_count = 4;         /* min repeat count */
734     /* tree[max_code+1].Len = -1; */  /* guard already set */
735     if (nextlen == 0) max_count = 138, min_count = 3;
737     for (n = 0; n <= max_code; n++) {
738         curlen = nextlen; nextlen = tree[n+1].Len;
739         if (++count < max_count && curlen == nextlen) {
740             continue;
741         } else if (count < min_count) {
742             do { send_code(s, curlen, s->bl_tree); } while (--count != 0);
744         } else if (curlen != 0) {
745             if (curlen != prevlen) {
746                 send_code(s, curlen, s->bl_tree); count--;
747             }
748             Assert(count >= 3 && count <= 6, " 3_6?");
749             send_code(s, REP_3_6, s->bl_tree); send_bits(s, count-3, 2);
751         } else if (count <= 10) {
752             send_code(s, REPZ_3_10, s->bl_tree); send_bits(s, count-3, 3);
754         } else {
755             send_code(s, REPZ_11_138, s->bl_tree); send_bits(s, count-11, 7);
756         }
757         count = 0; prevlen = curlen;
758         if (nextlen == 0) {
759             max_count = 138, min_count = 3;
760         } else if (curlen == nextlen) {
761             max_count = 6, min_count = 3;
762         } else {
763             max_count = 7, min_count = 4;
764         }
765     }
766 }
768 /* ===========================================================================
769  * Construct the Huffman tree for the bit lengths and return the index in
770  * bl_order of the last bit length code to send.
771  */
772 local int build_bl_tree(s)
773     deflate_state *s;
774 {
775     int max_blindex;  /* index of last bit length code of non zero freq */
777     /* Determine the bit length frequencies for literal and distance trees */
778     scan_tree(s, (ct_data *)s->dyn_ltree, s->l_desc.max_code);
779     scan_tree(s, (ct_data *)s->dyn_dtree, s->d_desc.max_code);
781     /* Build the bit length tree: */
782     build_tree(s, (tree_desc *)(&(s->bl_desc)));
783     /* opt_len now includes the length of the tree representations, except
784      * the lengths of the bit lengths codes and the 5+5+4 bits for the counts.
785      */
787     /* Determine the number of bit length codes to send. The pkzip format
```

```
788      * requires that at least 4 bit length codes be sent. (appnote.txt says
789      * 3 but the actual value used is 4.)
790      */
791     for (max_blindex = BL_CODES-1; max_blindex >= 3; max_blindex--) {
792         if (s->bl_tree[bl_order[max_blindex]].Len != 0) break;
793     }
794     /* Update opt_len to include the bit length tree and counts */
795     s->opt_len += 3*(max_blindex+1) + 5+5+4;
796     Tracev((stderr, "\ndyn trees: dyn %ld, stat %ld",
797             s->opt_len, s->static_len));
798
799     return max_blindex;
800 }
801
802 /* ===========================================================================
803  * Send the header for a block using dynamic Huffman trees: the counts, the
804  * lengths of the bit length codes, the literal tree and the distance tree.
805  * IN assertion: lcodes >= 257, dcodes >= 1, blcodes >= 4.
806  */
807 local void send_all_trees(s, lcodes, dcodes, blcodes)
808     deflate_state *s;
809     int lcodes, dcodes, blcodes; /* number of codes for each tree */
810 {
811     int rank;                    /* index in bl_order */
812
813     Assert (lcodes >= 257 && dcodes >= 1 && blcodes >= 4, "not enough codes");
814     Assert (lcodes <= L_CODES && dcodes <= D_CODES && blcodes <= BL_CODES,
815             "too many codes");
816     Tracev((stderr, "\nbl counts: "));
817     send_bits(s, lcodes-257, 5); /* not +255 as stated in appnote.txt */
818     send_bits(s, dcodes-1,   5);
819     send_bits(s, blcodes-4,  4); /* not -3 as stated in appnote.txt */
820     for (rank = 0; rank < blcodes; rank++) {
821         Tracev((stderr, "\nbl code %2d ", bl_order[rank]));
822         send_bits(s, s->bl_tree[bl_order[rank]].Len, 3);
823     }
824     Tracev((stderr, "\nbl tree: sent %ld", s->bits_sent));
825
826     send_tree(s, (ct_data *)s->dyn_ltree, lcodes-1); /* literal tree */
827     Tracev((stderr, "\nlit tree: sent %ld", s->bits_sent));
828
829     send_tree(s, (ct_data *)s->dyn_dtree, dcodes-1); /* distance tree */
830     Tracev((stderr, "\ndist tree: sent %ld", s->bits_sent));
831 }
832
833 /* ===========================================================================
834  * Send a stored block
835  */
836 void _tr_stored_block(s, buf, stored_len, eof)
837     deflate_state *s;
838     charf *buf;       /* input block */
839     ulg stored_len;   /* length of input block */
840     int eof;          /* true if this is the last block for a file */
841 {
842     send_bits(s, (STORED_BLOCK<<1)+eof, 3);  /* send block type */
843 #ifdef DEBUG
844     s->compressed_len = (s->compressed_len + 3 + 7) & (ulg)~7L;
845     s->compressed_len += (stored_len + 4) << 3;
846 #endif
847     copy_block(s, buf, (unsigned)stored_len, 1); /* with header */
848 }
849
850 /* ===========================================================================
851  * Send one empty static block to give enough lookahead for inflate.
852  * This takes 10 bits, of which 7 may remain in the bit buffer.
853  * The current inflate code requires 9 bits of lookahead. If the
```



```
854  * last two codes for the previous block (real code plus EOB) were coded
855  * on 5 bits or less, inflate may have only 5+3 bits of lookahead to decode
856  * the last real code. In this case we send two empty static blocks instead
857  * of one. (There are no problems if the previous block is stored or fixed.)
858  * To simplify the code, we assume the worst case of last real code encoded
859  * on one bit only.
860  */
861 void _tr_align(s)
862     deflate_state *s;
863 {
864     send_bits(s, STATIC_TREES<<1, 3);
865     send_code(s, END_BLOCK, static_ltree);
866 #ifdef DEBUG
867     s->compressed_len += 10L; /* 3 for block type, 7 for EOB */
868 #endif
869     bi_flush(s);
870     /* Of the 10 bits for the empty block, we have already sent
871      * (10 - bi_valid) bits. The lookahead for the last real code (before
872      * the EOB of the previous block) was thus at least one plus the length
873      * of the EOB plus what we have just sent of the empty static block.
874      */
875     if (1 + s->last_eob_len + 10 - s->bi_valid < 9) {
876         send_bits(s, STATIC_TREES<<1, 3);
877         send_code(s, END_BLOCK, static_ltree);
878 #ifdef DEBUG
879         s->compressed_len += 10L;
880 #endif
881         bi_flush(s);
882     }
883     s->last_eob_len = 7;
884 }
885
886 /* ===========================================================================
887  * Determine the best encoding for the current block: dynamic trees, static
888  * trees or store, and output the encoded block to the zip file.
889  */
890 void _tr_flush_block(s, buf, stored_len, eof)
891     deflate_state *s;
892     charf *buf;       /* input block, or NULL if too old */
893     ulg stored_len;   /* length of input block */
894     int eof;          /* true if this is the last block for a file */
895 {
896     ulg opt_lenb, static_lenb; /* opt_len and static_len in bytes */
897     int max_blindex = 0;  /* index of last bit length code of non zero freq */
898
899     /* Build the Huffman trees unless a stored block is forced */
900     if (s->level > 0) {
901
902          /* Check if the file is ascii or binary */
903         if (s->data_type == Z_UNKNOWN) set_data_type(s);
904
905         /* Construct the literal and distance trees */
906         build_tree(s, (tree_desc *)(&(s->l_desc)));
907         Tracev((stderr, "\nlit data: dyn %ld, stat %ld", s->opt_len,
908                 s->static_len));
909
910         build_tree(s, (tree_desc *)(&(s->d_desc)));
911         Tracev((stderr, "\ndist data: dyn %ld, stat %ld", s->opt_len,
912                 s->static_len));
913         /* At this point, opt_len and static_len are the total bit lengths of
914          * the compressed block data, excluding the tree representations.
915          */
916
917         /* Build the bit length tree for the above two trees, and get the index
918          * in bl_order of the last bit length code to send.
919          */
```

```
920         max_blindex = build_bl_tree(s);

922         /* Determine the best encoding. Compute first the block length in bytes*
923         opt_lenb = (s->opt_len+3+7)>>3;
924         static_lenb = (s->static_len+3+7)>>3;

926         Tracev((stderr, "\nopt %lu(%lu) stat %lu(%lu) stored %lu lit %u ",
927                 opt_lenb, s->opt_len, static_lenb, s->static_len, stored_len,
928                 s->last_lit));

930         if (static_lenb <= opt_lenb) opt_lenb = static_lenb;

932     } else {
933         Assert(buf != (char*)0, "lost buf");
934         opt_lenb = static_lenb = stored_len + 5; /* force a stored block */
935     }

937 #ifdef FORCE_STORED
938     if (buf != (char*)0) { /* force stored block */
939 #else
940     if (stored_len+4 <= opt_lenb && buf != (char*)0) {
941                         /* 4: two words for the lengths */
942 #endif
943         /* The test buf != NULL is only necessary if LIT_BUFSIZE > WSIZE.
944          * Otherwise we can't have processed more than WSIZE input bytes since
945          * the last block flush, because compression would have been
946          * successful. If LIT_BUFSIZE <= WSIZE, it is never too late to
947          * transform a block into a stored block.
948          */
949         _tr_stored_block(s, buf, stored_len, eof);

951 #ifdef FORCE_STATIC
952     } else if (static_lenb >= 0) { /* force static trees */
953 #else
954     } else if (static_lenb == opt_lenb) {
955 #endif
956         send_bits(s, (STATIC_TREES<<1)+eof, 3);
957         compress_block(s, (ct_data *)static_ltree, (ct_data *)static_dtree);
958 #ifdef DEBUG
959         s->compressed_len += 3 + s->static_len;
960 #endif
961     } else {
962         send_bits(s, (DYN_TREES<<1)+eof, 3);
963         send_all_trees(s, s->l_desc.max_code+1, s->d_desc.max_code+1,
964                        max_blindex+1);
965         compress_block(s, (ct_data *)s->dyn_ltree, (ct_data *)s->dyn_dtree);
966 #ifdef DEBUG
967         s->compressed_len += 3 + s->opt_len;
968 #endif
969     }
970     Assert (s->compressed_len == s->bits_sent, "bad compressed size");
971     /* The above check is made mod 2^32, for files larger than 512 MB
972      * and uLong implemented on 32 bits.
973      */
974     init_block(s);

976     if (eof) {
977         bi_windup(s);
978 #ifdef DEBUG
979         s->compressed_len += 7;  /* align on byte boundary */
980 #endif
981     }
982     Tracev((stderr,"\ncomprlen %lu(%lu) ", s->compressed_len>>3,
983            s->compressed_len-7*eof));
984 }
```



```
986 /* ===========================================================================
987  * Save the match info and tally the frequency counts. Return true if
988  * the current block must be flushed.
989  */
990 int _tr_tally (s, dist, lc)
991     deflate_state *s;
992     unsigned dist;  /* distance of matched string */
993     unsigned lc;    /* match length-MIN_MATCH or unmatched char (if dist==0) */
994 {
995     s->d_buf[s->last_lit] = (ush)dist;
996     s->l_buf[s->last_lit++] = (uch)lc;
997     if (dist == 0) {
998         /* lc is the unmatched char */
999         s->dyn_ltree[lc].Freq++;
1000    } else {
1001        s->matches++;
1002        /* Here, lc is the match length - MIN_MATCH */
1003        dist--;             /* dist = match distance - 1 */
1004        Assert((ush)dist < (ush)MAX_DIST(s) &&
1005               (ush)lc <= (ush)(MAX_MATCH-MIN_MATCH) &&
1006               (ush)d_code(dist) < (ush)D_CODES,  "_tr_tally: bad match");

1008        s->dyn_ltree[_length_code[lc]+LITERALS+1].Freq++;
1009        s->dyn_dtree[d_code(dist)].Freq++;
1010    }

1012 #ifdef TRUNCATE_BLOCK
1013    /* Try to guess if it is profitable to stop the current block here */
1014    if ((s->last_lit & 0x1fff) == 0 && s->level > 2) {
1015        /* Compute an upper bound for the compressed length */
1016        ulg out_length = (ulg)s->last_lit*8L;
1017        ulg in_length = (ulg)((long)s->strstart - s->block_start);
1018        int dcode;
1019        for (dcode = 0; dcode < D_CODES; dcode++) {
1020            out_length += (ulg)s->dyn_dtree[dcode].Freq *
1021                (5L+extra_dbits[dcode]);
1022        }
1023        out_length >>= 3;
1024        Tracev((stderr,"\nlast_lit %u, in %ld, out ~%ld(%ld%%) ",
1025               s->last_lit, in_length, out_length,
1026               100L - out_length*100L/in_length));
1027        if (s->matches < s->last_lit/2 && out_length < in_length/2) return 1;
1028    }
1029 #endif
1030    return (s->last_lit == s->lit_bufsize-1);
1031    /* We avoid equality with lit_bufsize because of wraparound at 64K
1032     * on 16 bit machines and because stored blocks are restricted to
1033     * 64K-1 bytes.
1034     */
1035 }

1037 /* ===========================================================================
1038  * Send the block data compressed using the given Huffman trees
1039  */
1040 local void compress_block(s, ltree, dtree)
1041     deflate_state *s;
1042     ct_data *ltree; /* literal tree */
1043     ct_data *dtree; /* distance tree */
1044 {
1045     unsigned dist;      /* distance of matched string */
1046     int lc;             /* match length or unmatched char (if dist == 0) */
1047     unsigned lx = 0;    /* running index in l_buf */
1048     unsigned code;      /* the code to send */
1049     int extra;          /* number of extra bits to send */

1051     if (s->last_lit != 0) do {
```

```
1052         dist = s->d_buf[lx];
1053         lc = s->l_buf[lx++];
1054         if (dist == 0) {
1055             send_code(s, lc, ltree); /* send a literal byte */
1056             Tracecv(isgraph(lc), (stderr," '%c' ", lc));
1057         } else {
1058             /* Here, lc is the match length - MIN_MATCH */
1059             code = _length_code[lc];
1060             send_code(s, code+LITERALS+1, ltree); /* send the length code */
1061             extra = extra_lbits[code];
1062             if (extra != 0) {
1063                 lc -= base_length[code];
1064                 send_bits(s, lc, extra);       /* send the extra length bits */
1065             }
1066             dist--; /* dist is now the match distance - 1 */
1067             code = d_code(dist);
1068             Assert (code < D_CODES, "bad d_code");

1070             send_code(s, code, dtree);       /* send the distance code */
1071             extra = extra_dbits[code];
1072             if (extra != 0) {
1073                 dist -= base_dist[code];
1074                 send_bits(s, dist, extra);   /* send the extra distance bits */
1075             }
1076         } /* literal or match pair ? */

1078         /* Check that the overlay between pending_buf and d_buf+l_buf is ok: */
1079         Assert(s->pending < s->lit_bufsize + 2*lx, "pendingBuf overflow");

1081     } while (lx < s->last_lit);

1083     send_code(s, END_BLOCK, ltree);
1084     s->last_eob_len = ltree[END_BLOCK].Len;
1085 }

1087 /* ===========================================================================
1088  * Set the data type to ASCII or BINARY, using a crude approximation:
1089  * binary if more than 20% of the bytes are <= 6 or >= 128, ascii otherwise.
1090  * IN assertion: the fields freq of dyn_ltree are set and the total of all
1091  * frequencies does not exceed 64K (to fit in an int on 16 bit machines).
1092  */
1093 local void set_data_type(s)
1094     deflate_state *s;
1095 {
1096     int n = 0;
1097     unsigned ascii_freq = 0;
1098     unsigned bin_freq = 0;
1099     while (n < 7)        bin_freq += s->dyn_ltree[n++].Freq;
1100     while (n < 128)    ascii_freq += s->dyn_ltree[n++].Freq;
1101     while (n < LITERALS) bin_freq += s->dyn_ltree[n++].Freq;
1102     s->data_type = (Byte)(bin_freq > (ascii_freq >> 2) ? Z_BINARY : Z_ASCII);
1103 }

1105 /* ===========================================================================
1106  * Reverse the first len bits of a code, using straightforward code (a faster
1107  * method would use a table)
1108  * IN assertion: 1 <= len <= 15
1109  */
1110 local unsigned bi_reverse(code, len)
1111     unsigned code; /* the value to invert */
1112     int len;       /* its bit length */
1113 {
1114     register unsigned res = 0;
1115     do {
1116         res |= code & 1;
1117         code >>= 1, res <<= 1;
```



```
1118     } while (--len > 0);
1119     return res >> 1;
1120 }

1122 /* ===========================================================================
1123  * Flush the bit buffer, keeping at most 7 bits in it.
1124  */
1125 local void bi_flush(s)
1126     deflate_state *s;
1127 {
1128     if (s->bi_valid == 16) {
1129         put_short(s, s->bi_buf);
1130         s->bi_buf = 0;
1131         s->bi_valid = 0;
1132     } else if (s->bi_valid >= 8) {
1133         put_byte(s, (Byte)s->bi_buf);
1134         s->bi_buf >>= 8;
1135         s->bi_valid -= 8;
1136     }
1137 }

1139 /* ===========================================================================
1140  * Flush the bit buffer and align the output on a byte boundary
1141  */
1142 local void bi_windup(s)
1143     deflate_state *s;
1144 {
1145     if (s->bi_valid > 8) {
1146         put_short(s, s->bi_buf);
1147     } else if (s->bi_valid > 0) {
1148         put_byte(s, (Byte)s->bi_buf);
1149     }
1150     s->bi_buf = 0;
1151     s->bi_valid = 0;
1152 #ifdef DEBUG
1153     s->bits_sent = (s->bits_sent+7) & ~7;
1154 #endif
1155 }

1157 /* ===========================================================================
1158  * Copy a stored block, storing first the length and its
1159  * one's complement if requested.
1160  */
1161 local void copy_block(s, buf, len, header)
1162     deflate_state *s;
1163     charf    *buf;    /* the input data */
1164     unsigned len;     /* its length */
1165     int      header;  /* true if block header must be written */
1166 {
1167     bi_windup(s);        /* align on byte boundary */
1168     s->last_eob_len = 8; /* enough lookahead for inflate */

1170     if (header) {
1171         put_short(s, (ush)len);
1172         put_short(s, (ush)~len);
1173 #ifdef DEBUG
1174         s->bits_sent += 2*16;
1175 #endif
1176     }
1177 #ifdef DEBUG
1178     s->bits_sent += (ulg)len<<3;
1179 #endif
1180     while (len--) {
1181         put_byte(s, *buf++);
1182     }
1183 }
```

```
**********************************************************
    8597 Wed Jan 31 21:21:18 2007
new/usr/src/uts/common/zmod/trees.h
1000001 add zlib compression support
**********************************************************
```

```
/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "@(#)trees.h    1.2     07/01/31 SMI"

/* header created automatically with -DGEN_TREES_H */

local const ct_data static_ltree[L_CODES+2] = {
{{ 12},{  8}}, {{140},{  8}}, {{ 76},{  8}}, {{204},{  8}}, {{ 44},{  8}},
{{172},{  8}}, {{108},{  8}}, {{236},{  8}}, {{ 28},{  8}}, {{156},{  8}},
{{ 92},{  8}}, {{220},{  8}}, {{ 60},{  8}}, {{188},{  8}}, {{124},{  8}},
{{252},{  8}}, {{  2},{  8}}, {{130},{  8}}, {{ 66},{  8}}, {{194},{  8}},
{{ 34},{  8}}, {{162},{  8}}, {{ 98},{  8}}, {{226},{  8}}, {{ 18},{  8}},
{{146},{  8}}, {{ 82},{  8}}, {{210},{  8}}, {{ 50},{  8}}, {{178},{  8}},
{{114},{  8}}, {{242},{  8}}, {{ 10},{  8}}, {{138},{  8}}, {{ 74},{  8}},
{{202},{  8}}, {{ 42},{  8}}, {{170},{  8}}, {{106},{  8}}, {{234},{  8}},
{{ 26},{  8}}, {{154},{  8}}, {{ 90},{  8}}, {{218},{  8}}, {{ 58},{  8}},
{{186},{  8}}, {{122},{  8}}, {{250},{  8}}, {{  6},{  8}}, {{134},{  8}},
{{ 70},{  8}}, {{198},{  8}}, {{ 38},{  8}}, {{166},{  8}}, {{102},{  8}},
{{230},{  8}}, {{ 22},{  8}}, {{150},{  8}}, {{ 86},{  8}}, {{214},{  8}},
{{ 54},{  8}}, {{182},{  8}}, {{118},{  8}}, {{246},{  8}}, {{ 14},{  8}},
{{142},{  8}}, {{ 78},{  8}}, {{206},{  8}}, {{ 46},{  8}}, {{174},{  8}},
{{110},{  8}}, {{238},{  8}}, {{ 30},{  8}}, {{158},{  8}}, {{ 94},{  8}},
{{222},{  8}}, {{ 62},{  8}}, {{190},{  8}}, {{126},{  8}}, {{254},{  8}},
{{  1},{  8}}, {{129},{  8}}, {{ 65},{  8}}, {{193},{  8}}, {{ 33},{  8}},
{{161},{  8}}, {{ 97},{  8}}, {{225},{  8}}, {{ 17},{  8}}, {{145},{  8}},
{{ 81},{  8}}, {{209},{  8}}, {{ 49},{  8}}, {{177},{  8}}, {{113},{  8}},
{{241},{  8}}, {{  9},{  8}}, {{137},{  8}}, {{ 73},{  8}}, {{201},{  8}},
{{ 41},{  8}}, {{169},{  8}}, {{105},{  8}}, {{233},{  8}}, {{ 25},{  8}},
{{153},{  8}}, {{ 89},{  8}}, {{217},{  8}}, {{ 57},{  8}}, {{185},{  8}},
{{121},{  8}}, {{249},{  8}}, {{  5},{  8}}, {{133},{  8}}, {{ 69},{  8}},
{{197},{  8}}, {{ 37},{  8}}, {{165},{  8}}, {{101},{  8}}, {{229},{  8}},
{{ 21},{  8}}, {{149},{  8}}, {{ 85},{  8}}, {{213},{  8}}, {{ 53},{  8}},
{{181},{  8}}, {{117},{  8}}, {{245},{  8}}, {{ 13},{  8}}, {{141},{  8}},
{{ 77},{  8}}, {{205},{  8}}, {{ 45},{  8}}, {{173},{  8}}, {{109},{  8}},
{{237},{  8}}, {{ 29},{  8}}, {{157},{  8}}, {{ 93},{  8}}, {{221},{  8}},
{{ 61},{  8}}, {{189},{  8}}, {{125},{  8}}, {{253},{  8}}, {{ 19},{  9}},
{{275},{  9}}, {{147},{  9}}, {{403},{  9}}, {{ 83},{  9}}, {{339},{  9}},
{{211},{  9}}, {{467},{  9}}, {{ 51},{  9}}, {{307},{  9}}, {{179},{  9}},
{{435},{  9}}, {{115},{  9}}, {{371},{  9}}, {{243},{  9}}, {{499},{  9}},
{{ 11},{  9}}, {{267},{  9}}, {{139},{  9}}, {{395},{  9}}, {{ 75},{  9}},
{{331},{  9}}, {{203},{  9}}, {{459},{  9}}, {{ 43},{  9}}, {{299},{  9}},
{{171},{  9}}, {{427},{  9}}, {{107},{  9}}, {{363},{  9}}, {{235},{  9}},
{{491},{  9}}, {{ 27},{  9}}, {{283},{  9}}, {{155},{  9}}, {{411},{  9}},
{{ 91},{  9}}, {{347},{  9}}, {{219},{  9}}, {{475},{  9}}, {{ 59},{  9}},
{{315},{  9}}, {{187},{  9}}, {{443},{  9}}, {{123},{  9}}, {{379},{  9}},
{{251},{  9}}, {{507},{  9}}, {{  7},{  9}}, {{263},{  9}}, {{135},{  9}},
{{391},{  9}}, {{ 71},{  9}}, {{327},{  9}}, {{199},{  9}}, {{455},{  9}},
{{ 39},{  9}}, {{295},{  9}}, {{167},{  9}}, {{423},{  9}}, {{103},{  9}},
{{359},{  9}}, {{231},{  9}}, {{487},{  9}}, {{ 23},{  9}}, {{279},{  9}},
{{151},{  9}}, {{407},{  9}}, {{ 87},{  9}}, {{343},{  9}}, {{215},{  9}},
{{471},{  9}}, {{ 55},{  9}}, {{311},{  9}}, {{183},{  9}}, {{439},{  9}},
{{119},{  9}}, {{375},{  9}}, {{247},{  9}}, {{503},{  9}}, {{ 15},{  9}},
{{271},{  9}}, {{143},{  9}}, {{399},{  9}}, {{ 79},{  9}}, {{335},{  9}},
{{207},{  9}}, {{463},{  9}}, {{ 47},{  9}}, {{303},{  9}}, {{175},{  9}},
{{431},{  9}}, {{111},{  9}}, {{367},{  9}}, {{239},{  9}}, {{495},{  9}},
{{ 31},{  9}}, {{287},{  9}}, {{159},{  9}}, {{415},{  9}}, {{ 95},{  9}},
{{351},{  9}}, {{223},{  9}}, {{479},{  9}}, {{ 63},{  9}}, {{319},{  9}},
{{191},{  9}}, {{447},{  9}}, {{127},{  9}}, {{383},{  9}}, {{255},{  9}},
```



```
{{511},{  9}}, {{  0},{  7}}, {{ 64},{  7}}, {{ 32},{  7}}, {{ 96},{  7}},
{{ 16},{  7}}, {{ 80},{  7}}, {{ 48},{  7}}, {{112},{  7}}, {{  8},{  7}},
{{ 72},{  7}}, {{ 40},{  7}}, {{104},{  7}}, {{ 24},{  7}}, {{ 88},{  7}},
{{ 56},{  7}}, {{120},{  7}}, {{  4},{  7}}, {{ 68},{  7}}, {{ 36},{  7}},
{{100},{  7}}, {{ 20},{  7}}, {{ 84},{  7}}, {{ 52},{  7}}, {{116},{  7}},
{{  3},{  8}}, {{131},{  8}}, {{ 67},{  8}}, {{195},{  8}}, {{ 35},{  8}},
{{163},{  8}}, {{ 99},{  8}}, {{227},{  8}}
};

local const ct_data static_dtree[D_CODES] = {
{{ 0},{ 5}}, {{16},{ 5}}, {{ 8},{ 5}}, {{24},{ 5}}, {{ 4},{ 5}},
{{20},{ 5}}, {{12},{ 5}}, {{28},{ 5}}, {{ 2},{ 5}}, {{18},{ 5}},
{{10},{ 5}}, {{26},{ 5}}, {{ 6},{ 5}}, {{22},{ 5}}, {{14},{ 5}},
{{30},{ 5}}, {{ 1},{ 5}}, {{17},{ 5}}, {{ 9},{ 5}}, {{25},{ 5}},
{{ 5},{ 5}}, {{21},{ 5}}, {{13},{ 5}}, {{29},{ 5}}, {{ 3},{ 5}},
{{19},{ 5}}, {{11},{ 5}}, {{27},{ 5}}, {{ 7},{ 5}}, {{23},{ 5}}
};

const uch _dist_code[DIST_CODE_LEN] = {
  0,  1,  2,  3,  4,  4,  5,  5,  6,  6,  6,  6,  7,  7,  7,  7,  8,  8,  8,  8,
  8,  8,  8,  8,  9,  9,  9,  9,  9,  9,  9,  9, 10, 10, 10, 10, 10, 10, 10, 10,
 10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13,
 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
 13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15,
 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,  0,  0, 16, 17,
 18, 18, 19, 19, 20, 20, 20, 20, 21, 21, 21, 21, 22, 22, 22, 22, 22, 22, 22, 22,
 23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
 24, 24, 24, 24, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27,
 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
 27, 27, 27, 27, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
 28, 28, 28, 28, 28, 28, 28, 28, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29
};

const uch _length_code[MAX_MATCH-MIN_MATCH+1]= {
  0,  1,  2,  3,  4,  5,  6,  7,  8,  8,  9, 10, 10, 11, 11, 12, 12, 12, 12,
 13, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 16,
 17, 17, 17, 17, 17, 17, 17, 17, 18, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19,
 19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22,
 22, 22, 22, 22, 22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 23, 23,
 23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26,
 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
 26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 28
};

local const int base_length[LENGTH_CODES] = {
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56,
64, 80, 96, 112, 128, 160, 192, 224, 0
```

```
128 };

130 local const int base_dist[D_CODES] = {
131     0,     1,     2,     3,     4,     6,     8,    12,    16,    24,
132    32,    48,    64,    96,   128,   192,   256,   384,   512,   768,
133  1024,  1536,  2048,  3072,  4096,  6144,  8192, 12288, 16384, 24576
134 };
```

```
**********************************************************
    3032 Wed Jan 31 21:21:18 2007
new/usr/src/uts/common/zmod/zconf.h
1000001 add zlib compression support
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Copyright 2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _ZCONF_H
#define _ZCONF_H

#pragma ident   "@(#)zconf.h    1.3     07/01/31 SMI"
#pragma ident   "@(#)zconf.h    1.2     05/07/18 SMI"

#include <sys/types.h>

#ifdef  __cplusplus
extern "C" {
#endif

#define z_off_t off_t
#define OF(p)   p
#define ZEXTERN extern
#define ZEXPORT
#define ZEXPORTVA
#define FAR

#define deflateInit_            z_deflateInit_
#define deflate                 z_deflate
#define deflateEnd              z_deflateEnd
#define inflateInit_            z_inflateInit_
#define inflate                 z_inflate
#define inflateEnd              z_inflateEnd
#define deflateInit2_           z_deflateInit2_
#define deflateSetDictionary    z_deflateSetDictionary
#define deflateCopy             z_deflateCopy
#define deflateReset            z_deflateReset
#define deflateParams           z_deflateParams
#define deflate_fast            z_deflate_fast
```



```
#define deflate_slow            z_deflate_slow
#define deflate_stored          z_deflate_stored
#define inflateInit2_           z_inflateInit2_
#define inflateSetDictionary    z_inflateSetDictionary
#define inflateSync             z_inflateSync
#define inflateSyncPoint        z_inflateSyncPoint
#define inflateReset            z_inflateReset
#define inflate_blocks          z_inflate_blocks
#define inflate_blocks_free     z_inflate_blocks_free
#define inflate_blocks_new      z_inflate_blocks_new
#define inflate_blocks_reset    z_inflate_blocks_reset
#define inflate_blocks_sync_point z_inflate_blocks_sync_point
#define inflate_codes           z_inflate_codes
#define inflate_codes_new       z_inflate_codes_new
#define inflate_codes_free      z_inflate_codes_free
#define inflate_fast            z_inflate_fast
#define inflate_flush           z_inflate_flush
#define inflate_mask            z_inflate_mask
#define inflate_trees_fixed     z_inflate_trees_fixed
#define inflate_trees_bits      z_inflate_trees_bits
#define inflate_trees_dynamic   z_inflate_trees_dynamic
#define inflate_set_dictionary  z_inflate_set_dictionary
#define adler32                 z_adler32
#define crc32                   z_crc32
#define get_crc_table           z_get_crc_table

#define MAX_MEM_LEVEL   9
#define MAX_WBITS       15

typedef unsigned char Byte;
typedef unsigned int uInt;
typedef unsigned long uLong;
typedef Byte Bytef;
typedef char charf;
typedef int intf;
typedef uInt uIntf;
typedef uLong uLongf;
typedef void *voidpf;
typedef void *voidp;

#ifdef  __cplusplus
}
```

_____*unchanged_portion_omitted_*

```
**********************************************************
    2976 Wed Jan 31 21:21:18 2007
new/usr/src/uts/common/zmod/zmod.c
1000001 add zlib compression support
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "@(#)zmod.c     1.4     07/01/31 SMI"
#pragma ident   "@(#)zmod.c     1.3     07/01/10 SMI"

#include <sys/modctl.h>
#include <sys/zmod.h>
#include <sys/systm.h>

#include "zlib.h"

/*
 * Uncompress the buffer 'src' into the buffer 'dst'.  The caller must store
 * the expected decompressed data size externally so it can be passed in.
 * The resulting decompressed size is then returned through dstlen.  This
 * function return Z_OK on success, or another error code on failure.
 */
int
z_uncompress(void *dst, size_t *dstlen, const void *src, size_t srclen)
{
        z_stream zs;
        int err;

        bzero(&zs, sizeof (zs));
        zs.next_in = (uchar_t *)src;
        zs.avail_in = srclen;
        zs.next_out = dst;
        zs.avail_out = *dstlen;

        if ((err = inflateInit(&zs)) != Z_OK)
                return (err);

        if ((err = inflate(&zs, Z_FINISH)) != Z_STREAM_END) {
                (void) inflateEnd(&zs);
                return (err == Z_OK ? Z_BUF_ERROR : err);
        }
```



```
        *dstlen = zs.total_out;
        return (inflateEnd(&zs));
}

int
z_compress(void *dst, size_t *dstlen, const void *src, size_t srclen)
{

        z_stream zs;
        int err;

        bzero(&zs, sizeof (zs));
        zs.next_in = (uchar_t *)src;
        zs.avail_in = srclen;
        zs.next_out = dst;
        zs.avail_out = *dstlen;

        if ((err = deflateInit(&zs, Z_DEFAULT_COMPRESSION)) != Z_OK)
                return (err);

        if ((err = deflate(&zs, Z_FINISH)) != Z_STREAM_END) {
                (void) deflateEnd(&zs);
                return (err == Z_OK ? Z_BUF_ERROR : err);
        }

        *dstlen = zs.total_out;
        return (deflateEnd(&zs));
}

static const char *const z_errmsg[] = {
        "need dictionary",      /* Z_NEED_DICT          2  */
        "stream end",           /* Z_STREAM_END         1  */
        "",                     /* Z_OK                 0  */
        "file error",           /* Z_ERRNO              (-1) */
        "stream error",         /* Z_STREAM_ERROR       (-2) */
        "data error",           /* Z_DATA_ERROR         (-3) */
        "insufficient memory",  /* Z_MEM_ERROR          (-4) */
        "buffer error",         /* Z_BUF_ERROR          (-5) */
        "incompatible version"  /* Z_VERSION_ERROR      (-6) */
};
```

*unchanged_portion_omitted_*

```
**********************************************************
    2225 Wed Jan 31 21:21:19 2007
new/usr/src/uts/common/zmod/zutil.c
1000001 add zlib compression support
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "@(#)zutil.c    1.4     07/01/31 SMI"
#pragma ident   "@(#)zutil.c    1.3     05/07/18 SMI"

#include <sys/systm.h>
#include <sys/cmn_err.h>
#include <sys/kobj.h>
#include <sys/kobj_impl.h>

/*
 * This module is used both during the normal operation of the kernel (i.e.
 * after kmem has been initialized) and during boot (before unix'_start has
 * been called).  kobj_alloc is able to tell the difference between the two
 * cases, and as such must be used instead of kmem_alloc.
 */

void
zmemcpy(uchar_t *dest, const uchar_t *source, uint_t len)
{
        bcopy(source, dest, len);
}

int
zmemcmp(const uchar_t *s1, const uchar_t *s2, uint_t len)
{
        return (bcmp(s1, s2, len));
}

void
zmemzero(uchar_t *dest, uint_t len)
{
        bzero(dest, len);
```



```
}

struct zchdr {
        uint_t zch_magic;
        uint_t zch_size;
};
______unchanged_portion_omitted_
```

```
**********************************************************
    1959 Wed Jan 31 21:21:19 2007
new/usr/src/uts/common/zmod/zutil.h
1000001 add zlib compression support
**********************************************************
```

```
   1 /*
   2  * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
   2  * Copyright 2003 Sun Microsystems, Inc.  All rights reserved.
   3  * Use is subject to license terms.
   4  */

   6 /*
   7  * zutil.h -- internal interface and configuration of the compression library
   8  * Copyright (C) 1995-1998 Jean-loup Gailly.
   9  * For conditions of distribution and use, see copyright notice in zlib.h
  10  */

  12 #ifndef _ZUTIL_H
  13 #define _ZUTIL_H

  15 #pragma ident   "@(#)zutil.h    1.2     07/01/31 SMI"
  15 #pragma ident   "@(#)zutil.h    1.1     03/09/02 SMI"

  17 #include <sys/types.h>

  19 #ifdef  __cplusplus
  20 extern "C" {
  21 #endif

  23 #include "zlib.h"

  25 #ifndef local
  26 #  define local static
  27 #endif

  29 typedef unsigned char  uch;
  30 typedef uch FAR uchf;
  31 typedef unsigned short ush;
  32 typedef ush FAR ushf;
  33 typedef unsigned long  ulg;

  35         /* common constants */

  37 #ifndef DEF_WBITS
  38 #  define DEF_WBITS MAX_WBITS
  39 #endif
  40 /* default windowBits for decompression. MAX_WBITS is for compression only */

  42 #if MAX_MEM_LEVEL >= 8
  43 #  define DEF_MEM_LEVEL 8
  44 #else
  45 #  define DEF_MEM_LEVEL  MAX_MEM_LEVEL
  46 #endif
  47 /* default memLevel */

  49 #define STORED_BLOCK 0
  50 #define STATIC_TREES 1
  51 #define DYN_TREES    2
  52 /* The three kinds of block type */

  54 #define MIN_MATCH  3
  55 #define MAX_MATCH  258
  56 /* The minimum and maximum match lengths */

  58 #define PRESET_DICT 0x20 /* preset dictionary flag in zlib header */
```



```
  60 #define Assert(cond,msg)
  61 #define Trace(x)
  62 #define Tracev(x)
  63 #define Tracevv(x)
  64 #define Tracec(c,x)
  65 #define Tracecv(c,x)

  67 typedef uLong (ZEXPORT *check_func)(uLong check, const Bytef *buf, uInt len);
  68 extern void zmemcpy(Bytef* dest, const Bytef* source, uInt len);
  69 extern int zmemcmp(const Bytef* s1, const Bytef* s2, uInt len);
  70 extern void zmemzero(Bytef* dest, uInt len);
  71 voidpf zcalloc(voidpf opaque, unsigned items, unsigned size);
  72 void   zcfree(voidpf opaque, voidpf ptr);

  74 #define ZALLOC(strm, items, size) \
  75           (*((strm)->zalloc))((strm)->opaque, (items), (size))
  76 #define ZFREE(strm, addr)  (*((strm)->zfree))((strm)->opaque, (voidpf)(addr))
  77 #define TRY_FREE(s, p) {if (p) ZFREE(s, p);}

  79 #ifdef  __cplusplus
  80 }
```

_____*unchanged_portion_omitted_*

```
**********************************************************
    6159 Wed Jan 31 21:21:19 2007
new/usr/src/uts/intel/Makefile.files
1000001 add zlib compression support
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# uts/intel/Makefile.files
#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)Makefile.files     1.39    07/01/31 SMI"
# ident "@(#)Makefile.files     1.38    07/01/21 SMI"
#
#       This Makefile defines all file modules and build rules for the
# directory uts/intel and its children. These are the source files which
# are specific to x86 processor architectures.
#

#
#       Core (unix) objects
#
CORE_OBJS +=            \
        arch_kdi.o      \
        copy.o          \
        copy_subr.o     \
        cpc_subr.o      \
        ddi_arch.o      \
        ddi_i86.o       \
        ddi_i86_asm.o   \
        desctbls.o      \
        desctbls_asm.o  \
        exception.o     \
        float.o         \
        fpu.o           \
        i86_subr.o      \
        lwp_private.o   \
        lock_prim.o     \
        ovbcopy.o       \
        polled_io.o     \
        sseblk.o        \
        sundep.o        \
        swtch.o         \
        sysi86.o
```



```
#
# 64-bit multiply/divide compiler helper routines
# used only for ia32
#

SPECIAL_OBJS_32 +=      \
        muldiv.o

#
#       Generic-unix Module
#
GENUNIX_OBJS +=         \
        archdep.o       \
        getcontext.o    \
        install_utrap.o \
        prom_enter.o    \
        prom_exit.o     \
        prom_panic.o    \
        sendsig.o       \
        syscall.o


#
#       PROM Routines
#
GENUNIX_OBJS +=         \
        prom_env.o      \
        prom_emul.o     \
        prom_getchar.o  \
        prom_init.o     \
        prom_node.o     \
        prom_printf.o   \
        prom_prop.o     \
        prom_putchar.o  \
        prom_reboot.o   \
        prom_version.o

#
#       file system modules
#
CORE_OBJS +=            \
        prmachdep.o

LX_PROC_OBJS += \
        lx_prsubr.o     \
        lx_prvfsops.o   \
        lx_prvnops.o

LX_AUTOFS_OBJS +=       \
        lx_autofs.o

#
#       Compression and decompression code
#       Decompression code
#
CORE_OBJS += decompress.o zutil.o zmod.o \
                adler32.o deflate.o infblock.o infcodes.o inffast.o \
                inflate.o inftrees.o infutil.o trees.o
CORE_OBJS += decompress.o adler32.o infblock.o infcodes.o inffast.o \
                inflate.o inftrees.o infutil.o zutil.o zmod.o

#
#       Driver modules
#
AGPGART_OBJS += agpgart.o agp_kstat.o
AGPTARGET_OBJS += agptarget.o
```

```
AMD64GART_OBJS += amd64_gart.o
ATA_OBJS += $(GHD_OBJS) ata_blacklist.o ata_common.o ata_disk.o \
        ata_dma.o atapi.o atapi_fsm.o ata_debug.o \
        sil3xxx.o
CMDK_OBJS += cmdk.o
CMLB_OBJS += cmlb.o
DADK_OBJS += dadk.o
DNET_OBJS += dnet.o mii.o
FD_OBJS += fd.o
GDA_OBJS += gda.o
GHD_OBJS += ghd.o ghd_debug.o ghd_dma.o ghd_queue.o ghd_scsa.o \
        ghd_scsi.o ghd_timer.o ghd_waitq.o ghd_gcmd.o
I915_OBJS += i915_sundrv.o i915_dma.o i915_drv.o i915_irq.o i915_mem.o
LOGI_OBJS += logi.o
MSMOUSE_OBJS += msm.o
MSCSI_OBJS += mscsi.o
PCICFG_OBJS += pcicfg.o
PCI_E_PCINEXUS_OBJS += pcie_pci.o
PCI_PCINEXUS_OBJS += pci_pci.o
PCIEHPCNEXUS_OBJS += pciehpc_x86.o pciehpc_acpi.o pciehpc_nvidia.o
PCN_OBJS += mii.o
POWER_OBJS += power.o
PCI_AUTOCONFIG_OBJS += pci_autoconfig.o pci_boot.o pcie_nvidia.o \
                       pci_memlist.o pci_resource.o
SD_OBJS += sd.o sd_xbuf.o
STRATEGY_OBJS += strategy.o
VGATEXT_OBJS += vgatext.o vgasubr.o

#
#       Kernel linker
#
KRTLD_OBJS +=           \
        bootrd.o        \
        ufsops.o        \
        hsfs.o          \
        doreloc.o       \
        kobj_boot.o     \
        kobj_convrelstr.o \
        kobj_crt.o      \
        kobj_isa.o      \
        kobj_reloc.o

#
#       misc. modules
#
ACPICA_OBJS     += dbcmds.o dbdisply.o \
                   dbexec.o dbfileio.o dbhistry.o dbinput.o dbstats.o \
                   dbutils.o dbxface.o evevent.o evgpe.o evgpeblk.o \
                   evmisc.o evregion.o evrgnini.o evsci.o evxface.o \
                   evxfevnt.o evxfregn.o hwacpi.o hwgpe.o hwregs.o \
                   hwsleep.o hwtimer.o dsfield.o dsinit.o dsmethod.o \
                   dsmthdat.o dsobject.o dsopcode.o dsutils.o dswexec.o \
                   dswload.o dswscope.o dswstate.o exconfig.o exconvrt.o \
                   excreate.o exdump.o exfield.o exfldio.o exmisc.o \
                   exmutex.o exnames.o exoparg1.o exoparg2.o exoparg3.o \
                   exoparg6.o exprep.o exregion.o exresnte.o exresolv.o \
                   exresop.o exstore.o exstoren.o exstorob.o exsystem.o \
                   exutils.o psargs.o psopcode.o psparse.o psscope.o \
                   pstree.o psutils.o pswalk.o psxface.o nsaccess.o \
                   nsalloc.o nsdump.o nsdumpdv.o nseval.o nsinit.o \
                   nsload.o nsnames.o nsobject.o nsparse.o nssearch.o \
                   nsutils.o nswalk.o nsxfeval.o nsxfname.o nsxfobj.o \
                   rsaddr.o rscalc.o rscreate.o rsdump.o \
                   rsinfo.o rsio.o rsirq.o rslist.o rsmemory.o rsmisc.o \
                   rsutils.o rsxface.o tbconvrt.o tbget.o tbgetall.o \
                   tbinstal.o tbrsdt.o tbutils.o tbxface.o tbxfroot.o \
```



```
                   utalloc.o utclib.o utcopy.o utdebug.o utdelete.o \
                   uteval.o utglobal.o utinit.o utmath.o utmisc.o \
                   utobject.o utresrc.o utxface.o acpica.o acpi_enum.o \
                   master_ops.o osl.o osl_ml.o acpica_ec.o utcache.o \
                   utmutex.o utstate.o dmbuffer.o dmnames.o dmobject.o \
                   dmopcode.o dmresrc.o dmresrcl.o dmresrcs.o dmutils.o \
                   dmwalk.o psloop.o uttrack.o

ACPI_INTP_OBJS += acpi_inf.o acpi_mod.o  acpi_ml.o \
        acpi_decl.o acpi_exe.o acpi_gram.o acpi_io.o acpi_lex.o \
        acpi_name.o acpi_ns.o acpi_op1.o acpi_op2.o acpi_rule.o \
        acpi_tab.o acpi_thr.o acpi_val.o \
        acpi_exc.o acpi_bst.o acpi_node.o acpi_stk.o acpi_par.o

AGP_OBJS += agpmaster.o
FBT_OBJS += fbt.o
SDT_OBJS += sdt.o

#
#       Pentium Performance Counter BackEnd module
#
P123_PCBE_OBJS = p123_pcbe.o

#
#       Pentium 4 Performance Counter BackEnd module
#
P4_PCBE_OBJS = p4_pcbe.o

#
#       AMD Opteron/Athlon64 Performance Counter BackEnd module
#
OPTERON_PCBE_OBJS = opteron_pcbe.o

#
#       AAC module
#
AAC_OBJS = aac.o aac_ioctl.o

#
#       AMR module
#
AMR_OBJS = amr.o

#
#       Brand modules
#
SN1_BRAND_OBJS  =       sn1_brand.o sn1_brand_asm.o

LX_BRAND_OBJS  =                \
        lx_brand.o              \
        lx_brand_asm.o          \
        lx_brk.o                \
        lx_clone.o              \
        lx_futex.o              \
        lx_getpid.o             \
        lx_id.o                 \
        lx_kill.o               \
        lx_misc.o               \
        lx_modify_ldt.o         \
        lx_pid.o                \
        lx_sched.o              \
        lx_signum.o             \
        lx_syscall.o            \
        lx_sysinfo.o            \
        lx_thread_area.o
```

```
256 #
257 #       special files
258 #
259 MODSTUB_OBJ +=          \
260         modstubs.o

262 BOOTDEV_OBJS +=         \
263         bootdev.o

265 INC_PATH        += -I$(UTSBASE)/intel
```

```
**********************************************************
    2458 Wed Jan 31 21:21:19 2007
new/usr/src/uts/intel/spppcomp/Makefile
1000001 add zlib compression support
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# uts/intel/spppcomp/Makefile
#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "@(#)Makefile   1.7     07/01/31 SMI"
#ident  "@(#)Makefile   1.6     06/11/02 SMI"
#
#       This makefile drives the production of the spppcomp STREAMS
#       kernel module.
#
#       intel architecture dependent
#


#
#       Path to the base of the uts directory tree (usually /usr/src/uts).
#
UTSBASE = ../..

#
#       Define the module and object file sets.
#
MODULE          = spppcomp
OBJECTS         = $(SPPPCOMP_OBJS:%=$(OBJS_DIR)/%)
LINTS           = $(SPPPCOMP_OBJS:%.o=$(LINTS_DIR)/%.ln)
ROOTMODULE      = $(USR_STRMOD_DIR)/$(MODULE)

#
#       Include common rules.
#
include $(UTSBASE)/intel/Makefile.intel

#
#       Define targets
#
ALL_TARGET      = $(BINARY)
LINT_TARGET     = $(MODULE).lint
INSTALL_TARGET  = $(BINARY) $(ROOTMODULE)
```



```
#
#       Internal build definitions
#
CPPFLAGS        += -DINTERNAL_BUILD -DSOL2 -DMUX_FRAME

#
#       Additional compiler definitions
#
INC_PATH        += -I$(UTSBASE)/common/io/ppp/common

#
# For now, disable these lint checks; maintainers should endeavor
# to investigate and remove these for maximum lint coverage.
# Please do not carry these forward to new Makefiles.
#
LINTTAGS        += -erroff=E_BAD_PTR_CAST_ALIGN
LINTTAGS        += -erroff=E_PTRDIFF_OVERFLOW
LINTTAGS        += -erroff=E_ASSIGN_NARROW_CONV

#
#       Default build targets.
#
.KEEP_STATE:

def:            $(DEF_DEPS)

all:            $(ALL_DEPS)

clean:          $(CLEAN_DEPS)

clobber:        $(CLOBBER_DEPS)

lint:           $(LINT_DEPS)

modlintlib:     $(MODLINTLIB_DEPS)

clean.lint:     $(CLEAN_LINT_DEPS)

install:        $(INSTALL_DEPS)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ppp/spppcomp/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

#
#       Include common targets.
#
include $(UTSBASE)/intel/Makefile.targ
```

```
**********************************************************
    2888 Wed Jan 31 21:21:20 2007
new/usr/src/uts/sparc/Makefile.files
1000001 add zlib compression support
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "@(#)Makefile.files     1.83    07/01/31 SMI"
#ident  "@(#)Makefile.files     1.82    07/01/10 SMI"
#
#       This Makefile defines all file modules and build rules for the
# directory uts/sparc and it's children. These are the source files which
# are specific to the sparc processor.
#

#
#       object lists
#

CORE_OBJS +=    ddi_arch.o      \
                polled_io.o     \
                sparc_ddi.o
#
#       decompression support
#
CORE_OBJS += adler32.o infblock.o infcodes.o inffast.o \
                inflate.o inftrees.o infutil.o zutil.o zmod.o


#
#       generic-unix module
#
GENUNIX_OBJS += addsub.o        \
                archdep.o       \
                bitmap_arch.o   \
                compare.o       \
                div.o           \
                fpu_simulator.o \
                getcontext.o    \
                iu_simulator.o  \
                mul.o           \
                pack.o          \
                sundep.o        \
                syscall.o       \
```



```
                unpack.o        \
                utility.o

#
#                       Driver (pseudo-driver) Modules
#

#
#                       Driver modules
#
FD_OBJS +=      fd_asm.o

CPR_SPARC_OBJS +=       cpr_sparc.o
PCI_PCI_OBJS +=         pci_pci.o pci_debug.o pci_pwr.o pcix.o
PX_PCI_OBJS +=          px_pci.o pcie_pwr.o
FCODE_OBJS +=           fcode.o

#
#                       file system modules
#
# XXX - currently a bug?...
#PROC_OBJS +=
CORE_OBJS +=    prmachdep.o

#
#                       misc modules
#
KRTLD_BOOT_OBJS +=      kobj_boot.o
KRTLD_OBJS +=           \
        bootops.o       \
        doreloc.o       \
        kobj_alloc.o    \
        kobj_convrelstr.o \
        kobj_crt.o      \
        kobj_isa.o      \
        kobj_reloc.o

SWAPGENERIC_OBJS += swapgeneric.o
PCICFG_E_OBJS   += pcicfg.e.o
FCPCI_OBJS      += fcpci.o
FCODEM_OBJS     += fc_ddi.o fc_physio.o fc_ops.o fc_subr.o

#
#                       special files
#
MODSTUB_OBJ =   modstubs.o

#
#                       SPARC DTrace Providers
#
FBT_OBJS        += fbt.o
SDT_OBJS        += sdt.o

#
#       Build up paths and defines.
#
LINT_DEFS       += -Dsparc
INC_PATH        += -I$(UTSBASE)/sparc

#
# Since assym.h is a derived file, the dependency must be explicit for
# all files including this file. (This is only actually required in the
# instance when the .nse_depinfo file does not exist.) It may seem that
# the lint targets should also have a similar dependency, but they don't
# since only C headers are included when #defined(lint) is true.
#
```

```
127 ASSYM_DEPS +=   sparc_ddi.o

129 #
130 # Inlined assembler routines.
131 #
132 INLINES         += $(UTSBASE)/sparc/ml/sparc.il
```

```
**********************************************************
    2525 Wed Jan 31 21:21:20 2007
new/usr/src/uts/sparc/spppcomp/Makefile
1000001 add zlib compression support
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# uts/sparc/spppcomp/Makefile
#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "@(#)Makefile   1.6     07/01/31 SMI"
#ident  "@(#)Makefile   1.5     06/11/02 SMI"
#
#       This makefile drives the production of the spppcomp STREAMS
#       kernel module.
#
#       sparc architecture dependent
#


#
#       Path to the base of the uts directory tree (usually /usr/src/uts).
#
UTSBASE = ../..

#
#       Define the module and object file sets.
#
MODULE          = spppcomp
OBJECTS         = $(SPPPCOMP_OBJS:%=$(OBJS_DIR)/%)
LINTS           = $(SPPPCOMP_OBJS:%.o=$(LINTS_DIR)/%.ln)
ROOTMODULE      = $(USR_STRMOD_DIR)/$(MODULE)

#
#       Include common rules.
#
include $(UTSBASE)/sparc/Makefile.sparc

#
#       Define targets
#
ALL_TARGET      = $(BINARY)
LINT_TARGET     = $(MODULE).lint
INSTALL_TARGET  = $(BINARY) $(ROOTMODULE)
```



```
#
#       Internal build definitions
#
CPPFLAGS        += -DINTERNAL_BUILD -DSOL2 -DMUX_FRAME

#
#       Additional compiler definitions
#
INC_PATH        += -I$(UTSBASE)/common/io/ppp/common
CFLAGS          += $(CCVERBOSE)

CLEANLINTFILES  += $(LINT64_FILES)

#
# For now, disable these lint checks; maintainers should endeavor
# to investigate and remove these for maximum lint coverage.
# Please do not carry these forward to new Makefiles.
#
LINTTAGS        += -erroff=E_BAD_PTR_CAST_ALIGN
LINTTAGS        += -erroff=E_PTRDIFF_OVERFLOW
LINTTAGS        += -erroff=E_ASSIGN_NARROW_CONV

#
#       Default build targets.
#
.KEEP_STATE:

def:            $(DEF_DEPS)

all:            $(ALL_DEPS)

clean:          $(CLEAN_DEPS)

clobber:        $(CLOBBER_DEPS)

lint:           $(LINT_DEPS)

modlintlib:     $(MODLINTLIB_DEPS) lint64

clean.lint:     $(CLEAN_LINT_DEPS)

install:        $(INSTALL_DEPS)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ppp/spppcomp/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

#
#       Include common targets.
#
include $(UTSBASE)/sparc/Makefile.targ
```