```
**********************************************************
     911 Fri Feb 21 02:07:10 2014
new/usr/src/cmd/perl/Makefile
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
#

include $(SRC)/cmd/Makefile.cmd

SUBDIRS = \
        contrib/Sun/Solaris/BSM \
        contrib/Sun/Solaris/Intrs \
        contrib/Sun/Solaris/Kstat \
        contrib/Sun/Solaris/Lgrp \
        contrib/Sun/Solaris/Pg \
        contrib/Sun/Solaris/Project \
        contrib/Sun/Solaris/Task \
        contrib/Sun/Solaris/Utils
# 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 (c) 2001, 2010, Oracle and/or its affiliates. All rights reserved.
#

include ../Makefile.cmd

all :=          TARGET = all
install :=      TARGET = install
clean :=        TARGET = clean
#endif /* ! codereview */
clobber :=      TARGET = clobber
clean   := TARGET = clean
test    := TARGET = test
```



```
# PERL_LEGACY is versions of Perl still delivered through ON
PERL_VERSIONS = 5.10.0

.PARALLEL: $(PERL_VERSIONS)

all install clean clobber: $(SUBDIRS)
all install test: $(PERL_VERSIONS)

$(SUBDIRS): FRC
        @cd $@; pwd; $(MAKE) $(TARGET)
clean: FRC

clobber: clean
        $(RM) -r $(PERL_VERSIONS)

#
# Perl is not lint-clean.  Fake up a target.
#
lint:
        @ $(ECHO) "usr/src/cmd/perl is not lint-clean: skipping"
        @ $(TRUE)

$(PERL_VERSIONS): FRC
        @ if [ ! -d $@ ]; then \
            $(CP) -r skel $@; \
        fi
        @ cd $@; pwd; PERL_VERSION=$@ $(MAKE) $(TARGET)

FRC:
```

```
**********************************************************
     611 Fri Feb 21 02:07:13 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Intrs/mapfile-vers
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
#

$mapfile_version 2

SYMBOL_SCOPE {
    global:
        boot_Sun__Solaris__Intrs;
        XS_Sun__Solaris__Intrs_is_apic;
        XS_Sun__Solaris__Intrs_intrmove;
    local:
        *;
};
#endif /* ! codereview */
```

```
**********************************************************
   48079 Fri Feb 21 02:07:13 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Kstat/Kstat.xs
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
/*
 * 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 (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2014 Racktop Systems.
#endif /* ! codereview */
 */

/*
 * Kstat.xs is a Perl XS (eXStension module) that makes the Solaris
 * kstat(3KSTAT) facility available to Perl scripts.  Kstat is a general-purpose
 * mechanism  for  providing kernel statistics to users.  The Solaris API is
 * function-based (see the manpage for details), but for ease of use in Perl
 * scripts this module presents the information as a nested hash data structure.
 * It would be too inefficient to read every kstat in the system, so this module
 * uses the Perl TIEHASH mechanism to implement a read-on-demand semantic, which
 * only reads and updates kstats as and when they are actually accessed.
 */

/*
 * Ignored raw kstats.
 *
 * Some raw kstats are ignored by this module, these are listed below.  The
 * most common reason is that the kstats are stored as arrays and the ks_ndata
 * and/or ks_data_size fields are invalid.  In this case it is impossible to
 * know how many records are in the array, so they can't be read.
 *
 * unix:*:sfmmu_percpu_stat
 * This is stored as an array with one entry per cpu.  Each element is of type
 * struct sfmmu_percpu_stat.  The ks_ndata and ks_data_size fields are bogus.
 *
 * ufs directio:*:UFS DirectIO Stats
 * The structure definition used for these kstats (ufs_directio_kstats) is in a
 * C file (uts/common/fs/ufs/ufs_directio.c) rather than a header file, so it
 * isn't accessible.
 *
 * qlc:*:statistics
 * This is a third-party driver for which we don't have source.
 *
```



```
 * mm:*:phys_installed
 * This is stored as an array of uint64_t, with each pair of values being the
 * (address, size) of a memory segment.  The ks_ndata and ks_data_size fields
 * are both zero.
 *
 * sockfs:*:sock_unix_list
 * This is stored as an array with one entry per active socket.  Each element
 * is of type struct k_sockinfo.  The ks_ndata and ks_data_size fields are both
 * zero.
 *
 * Note that the ks_ndata and ks_data_size of many non-array raw kstats are
 * also incorrect.  The relevant assertions are therefore commented out in the
 * appropriate raw kstat read routines.
 */

/* Kstat related includes */
#include <libgen.h>
#include <kstat.h>
#include <sys/var.h>
#include <sys/utsname.h>
#include <sys/sysinfo.h>
#include <sys/flock.h>
#include <sys/dnlc.h>
#include <nfs/nfs.h>
#include <nfs/nfs_clnt.h>

/* Ultra-specific kstat includes */
#ifdef __sparc
#include <vm/hat_sfmmu.h>       /* from /usr/platform/sun4u/include */
#include <sys/simmstat.h>       /* from /usr/platform/sun4u/include */
#include <sys/sysctrl.h>        /* from /usr/platform/sun4u/include */
#include <sys/fhc.h>            /* from /usr/include */
#endif

/*
 * Solaris #defines SP, which conflicts with the perl definition of SP
 * We don't need the Solaris one, so get rid of it to avoid warnings
 */
#undef SP

/* Perl XS includes */
#include "EXTERN.h"
#include "perl.h"
#include "XSUB.h"

/* Debug macros */
#define DEBUG_ID "Sun::Solaris::Kstat"
#ifdef KSTAT_DEBUG
#define PERL_ASSERT(EXP) \
    ((void)((EXP) || (croak("%s: assertion failed at %s:%d: %s", \
    DEBUG_ID, __FILE__, __LINE__, #EXP), 0), 0))
#define PERL_ASSERTMSG(EXP, MSG) \
    ((void)((EXP) || (croak(DEBUG_ID ": " MSG), 0), 0))
#else
#define PERL_ASSERT(EXP)                ((void)0)
#define PERL_ASSERTMSG(EXP, MSG)        ((void)0)
#endif

/* Macros for saving the contents of KSTAT_RAW structures */
#if defined(HAS_QUAD) && defined(USE_64_BIT_INT)
#define NEW_IV(V) \
    (newSViv((IVTYPE) V))
#define NEW_UV(V) \
    (newSVuv((UVTYPE) V))
#else
#define NEW_IV(V) \
```

```
    (V >= IV_MIN && V <= IV_MAX ? newSViv((IVTYPE) V) : newSVnv((NVTYPE) V))
#if defined(UVTYPE)
#define NEW_UV(V) \
    (V <= UV_MAX ? newSVuv((UVTYPE) V) : newSVnv((NVTYPE) V))
# else
#define NEW_UV(V) \
    (V <= IV_MAX ? newSViv((IVTYPE) V) : newSVnv((NVTYPE) V))
#endif
#endif
#define NEW_HRTIME(V) \
    newSVnv((NVTYPE) (V / 1000000000.0))

#define SAVE_FNP(H, F, K) \
    hv_store(H, K, sizeof (K) - 1, newSViv((IVTYPE)(uintptr_t)&F), 0)
#define SAVE_STRING(H, S, K, SS) \
    hv_store(H, #K, sizeof (#K) - 1, \
    newSVpvn(S->K, SS ? strlen(S->K) : sizeof(S->K)), 0)
#define SAVE_INT32(H, S, K) \
    hv_store(H, #K, sizeof (#K) - 1, NEW_IV(S->K), 0)
#define SAVE_UINT32(H, S, K) \
    hv_store(H, #K, sizeof (#K) - 1, NEW_UV(S->K), 0)
#define SAVE_INT64(H, S, K) \
    hv_store(H, #K, sizeof (#K) - 1, NEW_IV(S->K), 0)
#define SAVE_UINT64(H, S, K) \
    hv_store(H, #K, sizeof (#K) - 1, NEW_UV(S->K), 0)
#define SAVE_HRTIME(H, S, K) \
    hv_store(H, #K, sizeof (#K) - 1, NEW_HRTIME(S->K), 0)

/* Private structure used for saving kstat info in the tied hashes */
typedef struct {
        char            read;           /* Kstat block has been read before */
        char            valid;          /* Kstat still exists in kstat chain */
        char            strip_str;      /* Strip KSTAT_DATA_CHAR fields */
        kstat_ctl_t     *kstat_ctl;     /* Handle returned by kstat_open */
        kstat_t         *kstat;         /* Handle used by kstat_read */
} KstatInfo_t;

/* typedef for apply_to_ties callback functions */
typedef int (*ATTCb_t)(HV *, void *);

/* typedef for raw kstat reader functions */
typedef void (*kstat_raw_reader_t)(HV *, kstat_t *, int);

/* Hash of "module:name" to KSTAT_RAW read functions */
static HV *raw_kstat_lookup;

/*
 * Kstats come in two flavours, named and raw.  Raw kstats are just C structs,
 * so we need a function per raw kstat to convert the C struct into the
 * corresponding perl hash.  All such conversion functions are in the following
 * section.
 */

/*
 * Definitions in /usr/include/sys/cpuvar.h and /usr/include/sys/sysinfo.h
 */

static void
save_cpu_stat(HV *self, kstat_t *kp, int strip_str)
{
        cpu_stat_t    *statp;
        cpu_sysinfo_t *sysinfop;
        cpu_syswait_t *syswaitp;
        cpu_vminfo_t  *vminfop;

        /* PERL_ASSERT(kp->ks_ndata == 1); */
```



```
        PERL_ASSERT(kp->ks_data_size == sizeof (cpu_stat_t));
        statp = (cpu_stat_t *)(kp->ks_data);
        sysinfop = &statp->cpu_sysinfo;
        syswaitp = &statp->cpu_syswait;
        vminfop  = &statp->cpu_vminfo;

        hv_store(self, "idle", 4, NEW_UV(sysinfop->cpu[CPU_IDLE]), 0);
        hv_store(self, "user", 4, NEW_UV(sysinfop->cpu[CPU_USER]), 0);
        hv_store(self, "kernel", 6, NEW_UV(sysinfop->cpu[CPU_KERNEL]), 0);
        hv_store(self, "wait", 4, NEW_UV(sysinfop->cpu[CPU_WAIT]), 0);
        hv_store(self, "wait_io", 7, NEW_UV(sysinfop->wait[W_IO]), 0);
        hv_store(self, "wait_swap", 9, NEW_UV(sysinfop->wait[W_SWAP]), 0);
        hv_store(self, "wait_pio",  8, NEW_UV(sysinfop->wait[W_PIO]), 0);
        SAVE_UINT32(self, sysinfop, bread);
        SAVE_UINT32(self, sysinfop, bwrite);
        SAVE_UINT32(self, sysinfop, lread);
        SAVE_UINT32(self, sysinfop, lwrite);
        SAVE_UINT32(self, sysinfop, phread);
        SAVE_UINT32(self, sysinfop, phwrite);
        SAVE_UINT32(self, sysinfop, pswitch);
        SAVE_UINT32(self, sysinfop, trap);
        SAVE_UINT32(self, sysinfop, intr);
        SAVE_UINT32(self, sysinfop, syscall);
        SAVE_UINT32(self, sysinfop, sysread);
        SAVE_UINT32(self, sysinfop, syswrite);
        SAVE_UINT32(self, sysinfop, sysfork);
        SAVE_UINT32(self, sysinfop, sysvfork);
        SAVE_UINT32(self, sysinfop, sysexec);
        SAVE_UINT32(self, sysinfop, readch);
        SAVE_UINT32(self, sysinfop, writech);
        SAVE_UINT32(self, sysinfop, rcvint);
        SAVE_UINT32(self, sysinfop, xmtint);
        SAVE_UINT32(self, sysinfop, mdmint);
        SAVE_UINT32(self, sysinfop, rawch);
        SAVE_UINT32(self, sysinfop, canch);
        SAVE_UINT32(self, sysinfop, outch);
        SAVE_UINT32(self, sysinfop, msg);
        SAVE_UINT32(self, sysinfop, sema);
        SAVE_UINT32(self, sysinfop, namei);
        SAVE_UINT32(self, sysinfop, ufsiget);
        SAVE_UINT32(self, sysinfop, ufsdirblk);
        SAVE_UINT32(self, sysinfop, ufsipage);
        SAVE_UINT32(self, sysinfop, ufsinopage);
        SAVE_UINT32(self, sysinfop, inodeovf);
        SAVE_UINT32(self, sysinfop, fileovf);
        SAVE_UINT32(self, sysinfop, procovf);
        SAVE_UINT32(self, sysinfop, intrthread);
        SAVE_UINT32(self, sysinfop, intrblk);
        SAVE_UINT32(self, sysinfop, idlethread);
        SAVE_UINT32(self, sysinfop, inv_swtch);
        SAVE_UINT32(self, sysinfop, nthreads);
        SAVE_UINT32(self, sysinfop, cpumigrate);
        SAVE_UINT32(self, sysinfop, xcalls);
        SAVE_UINT32(self, sysinfop, mutex_adenters);
        SAVE_UINT32(self, sysinfop, rw_rdfails);
        SAVE_UINT32(self, sysinfop, rw_wrfails);
        SAVE_UINT32(self, sysinfop, modload);
        SAVE_UINT32(self, sysinfop, modunload);
        SAVE_UINT32(self, sysinfop, bawrite);
#ifdef STATISTICS       /* see header file */
        SAVE_UINT32(self, sysinfop, rw_enters);
        SAVE_UINT32(self, sysinfop, win_uo_cnt);
        SAVE_UINT32(self, sysinfop, win_uu_cnt);
        SAVE_UINT32(self, sysinfop, win_so_cnt);
        SAVE_UINT32(self, sysinfop, win_su_cnt);
        SAVE_UINT32(self, sysinfop, win_suo_cnt);
```

```
#endif

        SAVE_INT32(self, syswaitp, iowait);
        SAVE_INT32(self, syswaitp, swap);
        SAVE_INT32(self, syswaitp, physio);

        SAVE_UINT32(self, vminfop, pgrec);
        SAVE_UINT32(self, vminfop, pgfrec);
        SAVE_UINT32(self, vminfop, pgin);
        SAVE_UINT32(self, vminfop, pgpgin);
        SAVE_UINT32(self, vminfop, pgout);
        SAVE_UINT32(self, vminfop, pgpgout);
        SAVE_UINT32(self, vminfop, swapin);
        SAVE_UINT32(self, vminfop, pgswapin);
        SAVE_UINT32(self, vminfop, swapout);
        SAVE_UINT32(self, vminfop, pgswapout);
        SAVE_UINT32(self, vminfop, zfod);
        SAVE_UINT32(self, vminfop, dfree);
        SAVE_UINT32(self, vminfop, scan);
        SAVE_UINT32(self, vminfop, rev);
        SAVE_UINT32(self, vminfop, hat_fault);
        SAVE_UINT32(self, vminfop, as_fault);
        SAVE_UINT32(self, vminfop, maj_fault);
        SAVE_UINT32(self, vminfop, cow_fault);
        SAVE_UINT32(self, vminfop, prot_fault);
        SAVE_UINT32(self, vminfop, softlock);
        SAVE_UINT32(self, vminfop, kernel_asflt);
        SAVE_UINT32(self, vminfop, pgrrun);
        SAVE_UINT32(self, vminfop, execpgin);
        SAVE_UINT32(self, vminfop, execpgout);
        SAVE_UINT32(self, vminfop, execfree);
        SAVE_UINT32(self, vminfop, anonpgin);
        SAVE_UINT32(self, vminfop, anonpgout);
        SAVE_UINT32(self, vminfop, anonfree);
        SAVE_UINT32(self, vminfop, fspgin);
        SAVE_UINT32(self, vminfop, fspgout);
        SAVE_UINT32(self, vminfop, fsfree);
}

/*
 * Definitions in /usr/include/sys/var.h
 */

static void
save_var(HV *self, kstat_t *kp, int strip_str)
{
        struct var *varp;

        /* PERL_ASSERT(kp->ks_ndata == 1); */
        PERL_ASSERT(kp->ks_data_size == sizeof (struct var));
        varp = (struct var *)(kp->ks_data);

        SAVE_INT32(self, varp, v_buf);
        SAVE_INT32(self, varp, v_call);
        SAVE_INT32(self, varp, v_proc);
        SAVE_INT32(self, varp, v_maxupttl);
        SAVE_INT32(self, varp, v_nglobpris);
        SAVE_INT32(self, varp, v_maxsyspri);
        SAVE_INT32(self, varp, v_clist);
        SAVE_INT32(self, varp, v_maxup);
        SAVE_INT32(self, varp, v_hbuf);
        SAVE_INT32(self, varp, v_hmask);
        SAVE_INT32(self, varp, v_pbuf);
        SAVE_INT32(self, varp, v_sptmap);
        SAVE_INT32(self, varp, v_maxpmem);
        SAVE_INT32(self, varp, v_autoup);
```



```
        SAVE_INT32(self, varp, v_bufhwm);
}

/*
 * Definition in /usr/include/sys/dnlc.h
 */

static void
save_ncstats(HV *self, kstat_t *kp, int strip_str)
{
        struct ncstats *ncstatsp;

        /* PERL_ASSERT(kp->ks_ndata == 1); */
        PERL_ASSERT(kp->ks_data_size == sizeof (struct ncstats));
        ncstatsp = (struct ncstats *)(kp->ks_data);

        SAVE_INT32(self, ncstatsp, hits);
        SAVE_INT32(self, ncstatsp, misses);
        SAVE_INT32(self, ncstatsp, enters);
        SAVE_INT32(self, ncstatsp, dbl_enters);
        SAVE_INT32(self, ncstatsp, long_enter);
        SAVE_INT32(self, ncstatsp, long_look);
        SAVE_INT32(self, ncstatsp, move_to_front);
        SAVE_INT32(self, ncstatsp, purges);
}

/*
 * Definition in  /usr/include/sys/sysinfo.h
 */

static void
save_sysinfo(HV *self, kstat_t *kp, int strip_str)
{
        sysinfo_t *sysinfop;

        /* PERL_ASSERT(kp->ks_ndata == 1); */
        PERL_ASSERT(kp->ks_data_size == sizeof (sysinfo_t));
        sysinfop = (sysinfo_t *)(kp->ks_data);

        SAVE_UINT32(self, sysinfop, updates);
        SAVE_UINT32(self, sysinfop, runque);
        SAVE_UINT32(self, sysinfop, runocc);
        SAVE_UINT32(self, sysinfop, swpque);
        SAVE_UINT32(self, sysinfop, swpocc);
        SAVE_UINT32(self, sysinfop, waiting);
}

/*
 * Definition in  /usr/include/sys/sysinfo.h
 */

static void
save_vminfo(HV *self, kstat_t *kp, int strip_str)
{
        vminfo_t *vminfop;

        /* PERL_ASSERT(kp->ks_ndata == 1); */
        PERL_ASSERT(kp->ks_data_size == sizeof (vminfo_t));
        vminfop = (vminfo_t *)(kp->ks_data);

        SAVE_UINT64(self, vminfop, freemem);
        SAVE_UINT64(self, vminfop, swap_resv);
        SAVE_UINT64(self, vminfop, swap_alloc);
        SAVE_UINT64(self, vminfop, swap_avail);
        SAVE_UINT64(self, vminfop, swap_free);
        SAVE_UINT64(self, vminfop, updates);
```

```
}

/*
 * Definition in /usr/include/nfs/nfs_clnt.h
 */

static void
save_nfs(HV *self, kstat_t *kp, int strip_str)
{
        struct mntinfo_kstat *mntinfop;

        /* PERL_ASSERT(kp->ks_ndata == 1); */
        PERL_ASSERT(kp->ks_data_size == sizeof (struct mntinfo_kstat));
        mntinfop = (struct mntinfo_kstat *)(kp->ks_data);

        SAVE_STRING(self, mntinfop, mik_proto, strip_str);
        SAVE_UINT32(self, mntinfop, mik_vers);
        SAVE_UINT32(self, mntinfop, mik_flags);
        SAVE_UINT32(self, mntinfop, mik_secmod);
        SAVE_UINT32(self, mntinfop, mik_curread);
        SAVE_UINT32(self, mntinfop, mik_curwrite);
        SAVE_INT32(self, mntinfop, mik_timeo);
        SAVE_INT32(self, mntinfop, mik_retrans);
        SAVE_UINT32(self, mntinfop, mik_acregmin);
        SAVE_UINT32(self, mntinfop, mik_acregmax);
        SAVE_UINT32(self, mntinfop, mik_acdirmin);
        SAVE_UINT32(self, mntinfop, mik_acdirmax);
        hv_store(self, "lookup_srtt", 11,
            NEW_UV(mntinfop->mik_timers[0].srtt), 0);
        hv_store(self, "lookup_deviate", 14,
            NEW_UV(mntinfop->mik_timers[0].deviate), 0);
        hv_store(self, "lookup_rtxcur", 13,
            NEW_UV(mntinfop->mik_timers[0].rtxcur), 0);
        hv_store(self, "read_srtt", 9,
            NEW_UV(mntinfop->mik_timers[1].srtt), 0);
        hv_store(self, "read_deviate", 12,
            NEW_UV(mntinfop->mik_timers[1].deviate), 0);
        hv_store(self, "read_rtxcur", 11,
            NEW_UV(mntinfop->mik_timers[1].rtxcur), 0);
        hv_store(self, "write_srtt", 10,
            NEW_UV(mntinfop->mik_timers[2].srtt), 0);
        hv_store(self, "write_deviate", 13,
            NEW_UV(mntinfop->mik_timers[2].deviate), 0);
        hv_store(self, "write_rtxcur", 12,
            NEW_UV(mntinfop->mik_timers[2].rtxcur), 0);
        SAVE_UINT32(self, mntinfop, mik_noresponse);
        SAVE_UINT32(self, mntinfop, mik_failover);
        SAVE_UINT32(self, mntinfop, mik_remap);
        SAVE_STRING(self, mntinfop, mik_curserver, strip_str);
}

/*
 * The following struct => hash functions are all only present on the sparc
 * platform, so they are all conditionally compiled depending on __sparc
 */

/*
 * Definition in /usr/platform/sun4u/include/vm/hat_sfmmu.h
 */

#ifdef __sparc
static void
save_sfmmu_global_stat(HV *self, kstat_t *kp, int strip_str)
{
        struct sfmmu_global_stat *sfmmugp;
```



```
        /* PERL_ASSERT(kp->ks_ndata == 1); */
        PERL_ASSERT(kp->ks_data_size == sizeof (struct sfmmu_global_stat));
        sfmmugp = (struct sfmmu_global_stat *)(kp->ks_data);

        SAVE_INT32(self, sfmmugp, sf_tsb_exceptions);
        SAVE_INT32(self, sfmmugp, sf_tsb_raise_exception);
        SAVE_INT32(self, sfmmugp, sf_pagefaults);
        SAVE_INT32(self, sfmmugp, sf_uhash_searches);
        SAVE_INT32(self, sfmmugp, sf_uhash_links);
        SAVE_INT32(self, sfmmugp, sf_khash_searches);
        SAVE_INT32(self, sfmmugp, sf_khash_links);
        SAVE_INT32(self, sfmmugp, sf_swapout);
        SAVE_INT32(self, sfmmugp, sf_tsb_alloc);
        SAVE_INT32(self, sfmmugp, sf_tsb_allocfail);
        SAVE_INT32(self, sfmmugp, sf_tsb_sectsb_create);
        SAVE_INT32(self, sfmmugp, sf_scd_1sttsb_alloc);
        SAVE_INT32(self, sfmmugp, sf_scd_2ndtsb_alloc);
        SAVE_INT32(self, sfmmugp, sf_scd_1sttsb_allocfail);
        SAVE_INT32(self, sfmmugp, sf_scd_2ndtsb_allocfail);
        SAVE_INT32(self, sfmmugp, sf_tteload8k);
        SAVE_INT32(self, sfmmugp, sf_tteload64k);
        SAVE_INT32(self, sfmmugp, sf_tteload512k);
        SAVE_INT32(self, sfmmugp, sf_tteload4m);
        SAVE_INT32(self, sfmmugp, sf_tteload32m);
        SAVE_INT32(self, sfmmugp, sf_tteload256m);
        SAVE_INT32(self, sfmmugp, sf_tsb_load8k);
        SAVE_INT32(self, sfmmugp, sf_tsb_load4m);
        SAVE_INT32(self, sfmmugp, sf_hblk_hit);
        SAVE_INT32(self, sfmmugp, sf_hblk8_ncreate);
        SAVE_INT32(self, sfmmugp, sf_hblk8_nalloc);
        SAVE_INT32(self, sfmmugp, sf_hblk1_ncreate);
        SAVE_INT32(self, sfmmugp, sf_hblk1_nalloc);
        SAVE_INT32(self, sfmmugp, sf_hblk_slab_cnt);
        SAVE_INT32(self, sfmmugp, sf_hblk_reserve_cnt);
        SAVE_INT32(self, sfmmugp, sf_hblk_recurse_cnt);
        SAVE_INT32(self, sfmmugp, sf_hblk_reserve_hit);
        SAVE_INT32(self, sfmmugp, sf_get_free_success);
        SAVE_INT32(self, sfmmugp, sf_get_free_throttle);
        SAVE_INT32(self, sfmmugp, sf_get_free_fail);
        SAVE_INT32(self, sfmmugp, sf_put_free_success);
        SAVE_INT32(self, sfmmugp, sf_put_free_fail);
        SAVE_INT32(self, sfmmugp, sf_pgcolor_conflict);
        SAVE_INT32(self, sfmmugp, sf_uncache_conflict);
        SAVE_INT32(self, sfmmugp, sf_unload_conflict);
        SAVE_INT32(self, sfmmugp, sf_ism_uncache);
        SAVE_INT32(self, sfmmugp, sf_ism_recache);
        SAVE_INT32(self, sfmmugp, sf_recache);
        SAVE_INT32(self, sfmmugp, sf_steal_count);
        SAVE_INT32(self, sfmmugp, sf_pagesync);
        SAVE_INT32(self, sfmmugp, sf_clrwrt);
        SAVE_INT32(self, sfmmugp, sf_pagesync_invalid);
        SAVE_INT32(self, sfmmugp, sf_kernel_xcalls);
        SAVE_INT32(self, sfmmugp, sf_user_xcalls);
        SAVE_INT32(self, sfmmugp, sf_tsb_grow);
        SAVE_INT32(self, sfmmugp, sf_tsb_shrink);
        SAVE_INT32(self, sfmmugp, sf_tsb_resize_failures);
        SAVE_INT32(self, sfmmugp, sf_tsb_reloc);
        SAVE_INT32(self, sfmmugp, sf_user_vtop);
        SAVE_INT32(self, sfmmugp, sf_ctx_inv);
        SAVE_INT32(self, sfmmugp, sf_tlb_reprog_pgsz);
        SAVE_INT32(self, sfmmugp, sf_region_remap_demap);
        SAVE_INT32(self, sfmmugp, sf_create_scd);
        SAVE_INT32(self, sfmmugp, sf_join_scd);
        SAVE_INT32(self, sfmmugp, sf_leave_scd);
        SAVE_INT32(self, sfmmugp, sf_destroy_scd);
}
```

```
#endif

/*
 * Definition in /usr/platform/sun4u/include/vm/hat_sfmmu.h
 */

#ifdef __sparc
static void
save_sfmmu_tsbsize_stat(HV *self, kstat_t *kp, int strip_str)
{
        struct sfmmu_tsbsize_stat *sfmmutp;

        /* PERL_ASSERT(kp->ks_ndata == 1); */
        PERL_ASSERT(kp->ks_data_size == sizeof (struct sfmmu_tsbsize_stat));
        sfmmutp = (struct sfmmu_tsbsize_stat *)(kp->ks_data);

        SAVE_INT32(self, sfmmutp, sf_tsbsz_8k);
        SAVE_INT32(self, sfmmutp, sf_tsbsz_16k);
        SAVE_INT32(self, sfmmutp, sf_tsbsz_32k);
        SAVE_INT32(self, sfmmutp, sf_tsbsz_64k);
        SAVE_INT32(self, sfmmutp, sf_tsbsz_128k);
        SAVE_INT32(self, sfmmutp, sf_tsbsz_256k);
        SAVE_INT32(self, sfmmutp, sf_tsbsz_512k);
        SAVE_INT32(self, sfmmutp, sf_tsbsz_1m);
        SAVE_INT32(self, sfmmutp, sf_tsbsz_2m);
        SAVE_INT32(self, sfmmutp, sf_tsbsz_4m);
}
#endif

/*
 * Definition in /usr/platform/sun4u/include/sys/simmstat.h
 */

#ifdef __sparc
static void
save_simmstat(HV *self, kstat_t *kp, int strip_str)
{
        uchar_t *simmstatp;
        SV      *list;
        int     i;

        /* PERL_ASSERT(kp->ks_ndata == 1); */
        PERL_ASSERT(kp->ks_data_size == sizeof (uchar_t) * SIMM_COUNT);

        list = newSVpv("", 0);
        for (i = 0, simmstatp = (uchar_t *)(kp->ks_data);
        i < SIMM_COUNT - 1; i++, simmstatp++) {
                sv_catpvf(list, "%d,", *simmstatp);
        }
        sv_catpvf(list, "%d", *simmstatp);
        hv_store(self, "status", 6, list, 0);
}
#endif

/*
 * Used by save_temperature to make CSV lists from arrays of
 * short temperature values
 */

#ifdef __sparc
static SV *
short_array_to_SV(short *shortp, int len)
{
        SV  *list;

        list = newSVpv("", 0);
```



```
        for (; len > 1; len--, shortp++) {
                sv_catpvf(list, "%d,", *shortp);
        }
        sv_catpvf(list, "%d", *shortp);
        return (list);
}

/*
 * Definition in /usr/platform/sun4u/include/sys/fhc.h
 */

static void
save_temperature(HV *self, kstat_t *kp, int strip_str)
{
        struct temp_stats *tempsp;

        /* PERL_ASSERT(kp->ks_ndata == 1); */
        PERL_ASSERT(kp->ks_data_size == sizeof (struct temp_stats));
        tempsp = (struct temp_stats *)(kp->ks_data);

        SAVE_UINT32(self, tempsp, index);
        hv_store(self, "l1", 2, short_array_to_SV(tempsp->l1, L1_SZ), 0);
        hv_store(self, "l2", 2, short_array_to_SV(tempsp->l2, L2_SZ), 0);
        hv_store(self, "l3", 2, short_array_to_SV(tempsp->l3, L3_SZ), 0);
        hv_store(self, "l4", 2, short_array_to_SV(tempsp->l4, L4_SZ), 0);
        hv_store(self, "l5", 2, short_array_to_SV(tempsp->l5, L5_SZ), 0);
        SAVE_INT32(self, tempsp, max);
        SAVE_INT32(self, tempsp, min);
        SAVE_INT32(self, tempsp, state);
        SAVE_INT32(self, tempsp, temp_cnt);
        SAVE_INT32(self, tempsp, shutdown_cnt);
        SAVE_INT32(self, tempsp, version);
        SAVE_INT32(self, tempsp, trend);
        SAVE_INT32(self, tempsp, override);
}
#endif

/*
 * Not actually defined anywhere - just a short.  Yuck.
 */

#ifdef __sparc
static void
save_temp_over(HV *self, kstat_t *kp, int strip_str)
{
        short *shortp;

        /* PERL_ASSERT(kp->ks_ndata == 1); */
        PERL_ASSERT(kp->ks_data_size == sizeof (short));

        shortp = (short *)(kp->ks_data);
        hv_store(self, "override", 8, newSViv(*shortp), 0);
}
#endif

/*
 * Defined in /usr/platform/sun4u/include/sys/sysctrl.h
 * (Well, sort of.  Actually there's no structure, just a list of #defines
 * enumerating *some* of the array indexes.)
 */

#ifdef __sparc
static void
save_ps_shadow(HV *self, kstat_t *kp, int strip_str)
{
        uchar_t *ucharp;
```

```
654         /* PERL_ASSERT(kp->ks_ndata == 1); */
655         PERL_ASSERT(kp->ks_data_size == SYS_PS_COUNT);

657         ucharp = (uchar_t *)(kp->ks_data);
658         hv_store(self, "core_0", 6, newSViv(*ucharp++), 0);
659         hv_store(self, "core_1", 6, newSViv(*ucharp++), 0);
660         hv_store(self, "core_2", 6, newSViv(*ucharp++), 0);
661         hv_store(self, "core_3", 6, newSViv(*ucharp++), 0);
662         hv_store(self, "core_4", 6, newSViv(*ucharp++), 0);
663         hv_store(self, "core_5", 6, newSViv(*ucharp++), 0);
664         hv_store(self, "core_6", 6, newSViv(*ucharp++), 0);
665         hv_store(self, "core_7", 6, newSViv(*ucharp++), 0);
666         hv_store(self, "pps_0", 5, newSViv(*ucharp++), 0);
667         hv_store(self, "clk_33", 6, newSViv(*ucharp++), 0);
668         hv_store(self, "clk_50", 6, newSViv(*ucharp++), 0);
669         hv_store(self, "v5_p", 4, newSViv(*ucharp++), 0);
670         hv_store(self, "v12_p", 5, newSViv(*ucharp++), 0);
671         hv_store(self, "v5_aux", 6, newSViv(*ucharp++), 0);
672         hv_store(self, "v5_p_pch", 8, newSViv(*ucharp++), 0);
673         hv_store(self, "v12_p_pch", 9, newSViv(*ucharp++), 0);
674         hv_store(self, "v3_pch", 6, newSViv(*ucharp++), 0);
675         hv_store(self, "v5_pch", 6, newSViv(*ucharp++), 0);
676         hv_store(self, "p_fan", 5, newSViv(*ucharp++), 0);
677 }
678 #endif

680 /*
681  * Definition in /usr/platform/sun4u/include/sys/fhc.h
682  */

684 #ifdef __sparc
685 static void
686 save_fault_list(HV *self, kstat_t *kp, int strip_str)
687 {
688         struct ft_list  *faultp;
689         int             i;
690         char            name[KSTAT_STRLEN + 7]; /* room for 999999 faults */

692         /* PERL_ASSERT(kp->ks_ndata == 1); */
693         /* PERL_ASSERT(kp->ks_data_size == sizeof (struct ft_list)); */

695         for (i = 1, faultp = (struct ft_list *)(kp->ks_data);
696             i <= 999999 && i <= kp->ks_data_size / sizeof (struct ft_list);
697             i++, faultp++) {
698                 (void) snprintf(name, sizeof (name), "unit_%d", i);
699                 hv_store(self, name, strlen(name), newSViv(faultp->unit), 0);
700                 (void) snprintf(name, sizeof (name), "type_%d", i);
701                 hv_store(self, name, strlen(name), newSViv(faultp->type), 0);
702                 (void) snprintf(name, sizeof (name), "fclass_%d", i);
703                 hv_store(self, name, strlen(name), newSViv(faultp->fclass), 0);
704                 (void) snprintf(name, sizeof (name), "create_time_%d", i);
705                 hv_store(self, name, strlen(name),
706                     NEW_UV(faultp->create_time), 0);
707                 (void) snprintf(name, sizeof (name), "msg_%d", i);
708                 hv_store(self, name, strlen(name), newSVpv(faultp->msg, 0), 0);
709         }
710 }
711 #endif

713 /*
714  * We need to be able to find the function corresponding to a particular raw
715  * kstat.  To do this we ignore the instance and glue the module and name
716  * together to form a composite key.  We can then use the data in the kstat
717  * structure to find the appropriate function.  We use a perl hash to manage the
718  * lookup, where the key is "module:name" and the value is a pointer to the
```



```
719  * appropriate C function.
720  *
721  * Note that some kstats include the instance number as part of the module
722  * and/or name.  This could be construed as a bug.  However, to work around this
723  * we omit any digits from the module and name as we build the table in
724  * build_raw_kstat_loopup(), and we remove any digits from the module and name
725  * when we look up the functions in lookup_raw_kstat_fn()
726  */

728 /*
729  * This function is called when the XS is first dlopen()ed, and builds the
730  * lookup table as described above.
731  */

733 static void
734 build_raw_kstat_lookup()
735         {
736         /* Create new hash */
737         raw_kstat_lookup = newHV();

739         SAVE_FNP(raw_kstat_lookup, save_cpu_stat, "cpu_stat:cpu_stat");
740         SAVE_FNP(raw_kstat_lookup, save_var, "unix:var");
741         SAVE_FNP(raw_kstat_lookup, save_ncstats, "unix:ncstats");
742         SAVE_FNP(raw_kstat_lookup, save_sysinfo, "unix:sysinfo");
743         SAVE_FNP(raw_kstat_lookup, save_vminfo, "unix:vminfo");
744         SAVE_FNP(raw_kstat_lookup, save_nfs, "nfs:mntinfo");
745 #ifdef __sparc
746         SAVE_FNP(raw_kstat_lookup, save_sfmmu_global_stat,
747             "unix:sfmmu_global_stat");
748         SAVE_FNP(raw_kstat_lookup, save_sfmmu_tsbsize_stat,
749             "unix:sfmmu_tsbsize_stat");
750         SAVE_FNP(raw_kstat_lookup, save_simmstat, "unix:simm-status");
751         SAVE_FNP(raw_kstat_lookup, save_temperature, "unix:temperature");
752         SAVE_FNP(raw_kstat_lookup, save_temp_over, "unix:temperature override");
753         SAVE_FNP(raw_kstat_lookup, save_ps_shadow, "unix:ps_shadow");
754         SAVE_FNP(raw_kstat_lookup, save_fault_list, "unix:fault_list");
755 #endif
756 }

758 /*
759  * This finds and returns the raw kstat reader function corresponding to the
760  * supplied module and name.  If no matching function exists, 0 is returned.
761  */

763 static kstat_raw_reader_t lookup_raw_kstat_fn(char *module, char *name)
764         {
765         char                    key[KSTAT_STRLEN * 2];
766         register char           *f, *t;
767         SV                      **entry;
768         kstat_raw_reader_t      fnp;

770         /* Copy across module & name, removing any digits - see comment above */
771         for (f = module, t = key; *f != '\0'; f++, t++) {
772                 while (*f != '\0' && isdigit(*f)) { f++; }
773                 *t = *f;
774         }
775         *t++ = ':';
776         for (f = name; *f != '\0'; f++, t++) {
777                 while (*f != '\0' && isdigit(*f)) {
778                         f++;
779                 }
780         *t = *f;
781         }
782         *t = '\0';

784         /* look up & return the function, or teturn 0 if not found */
```

```
785         if ((entry = hv_fetch(raw_kstat_lookup, key, strlen(key), FALSE)) == 0)
786         {
787                 fnp = 0;
788         } else {
789                 fnp = (kstat_raw_reader_t)(uintptr_t)SvIV(*entry);
790         }
791         return (fnp);
792 }

794 /*
795  * This module converts the flat list returned by kstat_read() into a perl hash
796  * tree keyed on module, instance, name and statistic.  The following functions
797  * provide code to create the nested hashes, and to iterate over them.
798  */

800 /*
801  * Given module, instance and name keys return a pointer to the hash tied to
802  * the bottommost hash.  If the hash already exists, we just return a pointer
803  * to it, otherwise we create the hash and any others also required above it in
804  * the hierarchy.  The returned tiehash is blessed into the
805  * Sun::Solaris::Kstat::_Stat class, so that the appropriate TIEHASH methods are
806  * called when the bottommost hash is accessed.  If the is_new parameter is
807  * non-null it will be set to TRUE if a new tie has been created, and FALSE if
808  * the tie already existed.
809  */

811 static HV *
812 get_tie(SV *self, char *module, int instance, char *name, int *is_new)
813 {
814         char str_inst[11];      /* big enough for up to 10^10 instances */
815         char *key[3];           /* 3 part key: module, instance, name */
816         int  k;
817         int  new;
818         HV   *hash;
819         HV   *tie;

821         /* Create the keys */
822         (void) snprintf(str_inst, sizeof (str_inst), "%d", instance);
823         key[0] = module;
824         key[1] = str_inst;
825         key[2] = name;

827         /* Iteratively descend the tree, creating new hashes as required */
828         hash = (HV *)SvRV(self);
829         for (k = 0; k < 3; k++) {
830                 SV **entry;

832                 SvREADONLY_off(hash);
833                 entry = hv_fetch(hash, key[k], strlen(key[k]), TRUE);

835                 /* If the entry doesn't exist, create it */
836                 if (! SvOK(*entry)) {
837                         HV *newhash;
838                         SV *rv;

840                         newhash = newHV();
841                         rv = newRV_noinc((SV *)newhash);
842                         sv_setsv(*entry, rv);
843                         SvREFCNT_dec(rv);
844                         if (k < 2) {
845                                 SvREADONLY_on(newhash);
846                         }
847                         SvREADONLY_on(*entry);
848                         SvREADONLY_on(hash);
849                         hash = newhash;
850                         new = 1;
```



```
852                 /* Otherwise it already existed */
853                 } else {
854                         SvREADONLY_on(hash);
855                         hash = (HV *)SvRV(*entry);
856                         new = 0;
857                 }
858         }

860         /* Create and bless a hash for the tie, if necessary */
861         if (new) {
862                 SV *tieref;
863                 HV *stash;

865                 tie = newHV();
866                 tieref = newRV_noinc((SV *)tie);
867                 stash = gv_stashpv("Sun::Solaris::Kstat::_Stat", TRUE);
868                 sv_bless(tieref, stash);

870                 /* Add TIEHASH magic */
871                 hv_magic(hash, (GV *)tieref, 'P');
872                 SvREADONLY_on(hash);

874         /* Otherwise, just find the existing tied hash */
875         } else {
876                 MAGIC *mg;

878                 mg = mg_find((SV *)hash, 'P');
879                 PERL_ASSERTMSG(mg != 0, "get_tie: lost P magic");
880                 tie = (HV *)SvRV(mg->mg_obj);
881         }
882         if (is_new) {
883                 *is_new = new;
884         }
885         return (tie);
886 }

888 /*
889  * This is an iterator function used to traverse the hash hierarchy and apply
890  * the passed function to the tied hashes at the bottom of the hierarchy.  If
891  * any of the callback functions return 0, 0 is returned, otherwise 1
892  */

894 static int
895 apply_to_ties(SV *self, ATTCb_t cb, void *arg)
896 {
897         HV      *hash1;
898         HE      *entry1;
 24         long    s;
899         int     ret;

901         hash1 = (HV *)SvRV(self);
902         hv_iterinit(hash1);
903         ret = 1;

905         /* Iterate over each module */
906         while ((entry1 = hv_iternext(hash1))) {
 32         while (entry1 = hv_iternext(hash1)) {
907                 HV *hash2;
908                 HE *entry2;

910                 hash2 = (HV *)SvRV(hv_iterval(hash1, entry1));
911                 hv_iterinit(hash2);

913                 /* Iterate over each module:instance */
914                 while ((entry2 = hv_iternext(hash2))) {
```

```
 40                 while (entry2 = hv_iternext(hash2)) {
915                         HV *hash3;
916                         HE *entry3;

918                         hash3 = (HV *)SvRV(hv_interval(hash2, entry2));
919                         hv_iterinit(hash3);

921                         /* Iterate over each module:instance:name */
922                         while ((entry3 = hv_iternext(hash3))) {
 48                         while (entry3 = hv_iternext(hash3)) {
923                                 HV    *hash4;
924                                 MAGIC *mg;
 51                                 HV    *tie;

926                                 /* Get the tie */
927                                 hash4 = (HV *)SvRV(hv_interval(hash3, entry3));
928                                 mg = mg_find((SV *)hash4, 'P');
929                                 PERL_ASSERTMSG(mg != 0,
930                                     "apply_to_ties: lost P magic");

932                                 /* Apply the callback */
933                                 if (! cb((HV *)SvRV(mg->mg_obj), arg)) {
934                                         ret = 0;
935                                 }
936                         }
937                 }
938         }
939         return (ret);
940 }
_____unchanged_portion_omitted_

957 /*
958  * Prune invalid kstat nodes. This is called when kstat_chain_update() detects
959  * that the kstat chain has been updated.  This removes any hash tree entries
960  * that no longer have a corresponding kstat.  If del is non-null it will be
961  * set to the keys of the deleted kstat nodes, if any.  If any entries are
962  * deleted 1 will be retured, otherwise 0
963  */

965 static int
966 prune_invalid(SV *self, AV *del)
967 {
968         HV      *hash1;
969         HE      *entry1;
970         STRLEN  klen;
971         char    *module, *instance, *name, *key;
972         int     ret;

974         hash1 = (HV *)SvRV(self);
975         hv_iterinit(hash1);
976         ret = 0;

978         /* Iterate over each module */
979         while ((entry1 = hv_iternext(hash1))) {
106         while (entry1 = hv_iternext(hash1)) {
980                 HV *hash2;
981                 HE *entry2;

983                 module = HePV(entry1, PL_na);
984                 hash2 = (HV *)SvRV(hv_interval(hash1, entry1));
985                 hv_iterinit(hash2);

987                 /* Iterate over each module:instance */
988                 while ((entry2 = hv_iternext(hash2))) {
115                 while (entry2 = hv_iternext(hash2)) {
989                         HV *hash3;
```



```
990                         HE *entry3;

992                         instance = HePV(entry2, PL_na);
993                         hash3 = (HV *)SvRV(hv_interval(hash2, entry2));
994                         hv_iterinit(hash3);

996                         /* Iterate over each module:instance:name */
997                         while ((entry3 = hv_iternext(hash3))) {
124                         while (entry3 = hv_iternext(hash3)) {
998                                 HV    *hash4;
999                                 MAGIC *mg;
1000                                HV    *tie;

1002                                name = HePV(entry3, PL_na);
1003                                hash4 = (HV *)SvRV(hv_interval(hash3, entry3));
1004                                mg = mg_find((SV *)hash4, 'P');
1005                                PERL_ASSERTMSG(mg != 0,
1006                                    "prune_invalid: lost P magic");
1007                                tie = (HV *)SvRV(mg->mg_obj);
1008                                mg = mg_find((SV *)tie, '~');
1009                                PERL_ASSERTMSG(mg != 0,
1010                                    "prune_invalid: lost ~ magic");

1012                                /* If this is marked as invalid, prune it */
1013                                if (((KstatInfo_t *)SvPVX(
1014                                    (SV *)mg->mg_obj))->valid == FALSE) {
1015                                        SvREADONLY_off(hash3);
1016                                        key = HePV(entry3, klen);
1017                                        hv_delete(hash3, key, klen, G_DISCARD);
1018                                        SvREADONLY_on(hash3);
1019                                        if (del) {
1020                                                av_push(del,
1021                                                    newSVpvf("%s:%s:%s",
1022                                                    module, instance, name));
1023                                        }
1024                                        ret = 1;
1025                                }
1026                        }

1028                        /* If the module:instance:name hash is empty prune it */
1029                        if (HvKEYS(hash3) == 0) {
1030                                SvREADONLY_off(hash2);
1031                                key = HePV(entry2, klen);
1032                                hv_delete(hash2, key, klen, G_DISCARD);
1033                                SvREADONLY_on(hash2);
1034                        }
1035                }
1036                /* If the module:instance hash is empty prune it */
1037                if (HvKEYS(hash2) == 0) {
1038                        SvREADONLY_off(hash1);
1039                        key = HePV(entry1, klen);
1040                        hv_delete(hash1, key, klen, G_DISCARD);
1041                        SvREADONLY_on(hash1);
1042                }
1043        }
1044        return (ret);
1045 }

1047 /*
1048  * Named kstats are returned as a list of key/values.  This function converts
1049  * such a list into the equivalent perl datatypes, and stores them in the passed
1050  * hash.
1051  */

1053 static void
1054 save_named(HV *self, kstat_t *kp, int strip_str)
```

```
1055 {
1056         kstat_named_t   *knp;
1057         int             n;
1058         SV*             value;

1060         for (n = kp->ks_ndata, knp = KSTAT_NAMED_PTR(kp); n > 0; n--, knp++) {
1061                 switch (knp->data_type) {
1062                 case KSTAT_DATA_CHAR:
1063                         value = newSVpv(knp->value.c, strip_str ?
1064                             strlen(knp->value.c) : sizeof (knp->value.c));
1065                         break;
1066                 case KSTAT_DATA_INT32:
1067                         value = newSViv(knp->value.i32);
1068                         break;
1069                 case KSTAT_DATA_UINT32:
1070                         value = NEW_UV(knp->value.ui32);
1071                         break;
1072                 case KSTAT_DATA_INT64:
1073                         value = NEW_UV(knp->value.i64);
1074                         break;
1075                 case KSTAT_DATA_UINT64:
1076                         value = NEW_UV(knp->value.ui64);
1077                         break;
1078                 case KSTAT_DATA_STRING:
1079                         if (KSTAT_NAMED_STR_PTR(knp) == NULL)
1080                                 value = newSVpv("null", sizeof ("null") - 1);
1081                         else
1082                                 value = newSVpv(KSTAT_NAMED_STR_PTR(knp),
1083                                                 KSTAT_NAMED_STR_BUFLEN(knp) -1);
1084                         break;
1085                 default:
1086                         PERL_ASSERTMSG(0, "kstat_read: invalid data type");
1087                         continue;
 214                         break;
1088                 }
1089                 hv_store(self, knp->name, strlen(knp->name), value, 0);
1090         }
1091 }
```

*unchanged_portion_omitted_*

```
1223 /*
1224  * The XS code exported to perl is below here.  Note that the XS preprocessor
1225  * has its own commenting syntax, so all comments from this point on are in
1226  * that form.
1227  */

1229 /* The following XS methods are the ABI of the Sun::Solaris::Kstat package */

1231 MODULE = Sun::Solaris::Kstat PACKAGE = Sun::Solaris::Kstat
1232 PROTOTYPES: ENABLE

1234  # Create the raw kstat to store function lookup table on load
1235 BOOT:
1236         build_raw_kstat_lookup();

1238  #
1239  # The Sun::Solaris::Kstat constructor.  This builds the nested
1240  # name::instance::module hash structure, but doesn't actually read the
1241  # underlying kstats.  This is done on demand by the TIEHASH methods in
1242  # Sun::Solaris::Kstat::_Stat
1243  #

1245 SV*
1246 new(class, ...)
1247         char *class;
1248 PREINIT:
```



```
1249         HV              *stash;
1250         kstat_ctl_t     *kc;
1251         SV              *kcsv;
1252         kstat_t         *kp;
1253         KstatInfo_t     kstatinfo;
1254         int             sp, strip_str;
1255 CODE:
1256         /* Check we have an even number of arguments, excluding the class */
1257         sp = 1;
1258         if (((items - sp) % 2) != 0) {
1259                 croak(DEBUG_ID ": new: invalid number of arguments");
1260         }

1262         /* Process any (name => value) arguments */
1263         strip_str = 0;
1264         while (sp < items) {
1265                 SV *name, *value;

1267                 name = ST(sp);
1268                 sp++;
1269                 value = ST(sp);
1270                 sp++;
1271                 if (strcmp(SvPVX(name), "strip_strings") == 0) {
1272                         strip_str = SvTRUE(value);
1273                 } else {
1274                         croak(DEBUG_ID ": new: invalid parameter name '%s'",
1275                             SvPVX(name));
1276                 }
1277         }

1279         /* Open the kstats handle */
1280         if ((kc = kstat_open()) == 0) {
1281                 XSRETURN_UNDEF;
1282         }

1284         /* Create a blessed hash ref */
1285         RETVAL = (SV *)newRV_noinc((SV *)newHV());
1286         stash = gv_stashpv(class, TRUE);
1287         sv_bless(RETVAL, stash);

1289         /* Create a place to save the KstatInfo_t structure */
1290         kcsv = newSVpv((char *)&kc, sizeof (kc));
1291         sv_magic(SvRV(RETVAL), kcsv, '~', 0, 0);
1292         SvREFCNT_dec(kcsv);

1294         /* Initialise the KstatsInfo_t structure */
1295         kstatinfo.read = FALSE;
1296         kstatinfo.valid = TRUE;
1297         kstatinfo.strip_str = strip_str;
1298         kstatinfo.kstat_ctl = kc;

1300         /* Scan the kstat chain, building hash entries for the kstats */
1301         for (kp = kc->kc_chain; kp != 0; kp = kp->ks_next) {
1302                 HV *tie;
1303                 SV *kstatsv;

1305                 /* Don't bother storing the kstat headers */
1306                 if (strncmp(kp->ks_name, "kstat_", 6) == 0) {
1307                         continue;
1308                 }

1310                 /* Don't bother storing raw stats we don't understand */
1311                 if (kp->ks_type == KSTAT_TYPE_RAW &&
1312                     lookup_raw_kstat_fn(kp->ks_module, kp->ks_name) == 0) {
1313 #ifdef REPORT_UNKNOWN
1314                         (void) fprintf(stderr,
```

```
1315                            "Unknown kstat type %s:%d:%s - %d of size %d\n",
1316                            kp->ks_module, kp->ks_instance, kp->ks_name,
1317                            kp->ks_ndata, kp->ks_data_size);
1318 #endif
1319                         continue;
1320                 }

1322                 /* Create a 3-layer hash hierarchy - module.instance.name */
1323                 tie = get_tie(RETVAL, kp->ks_module, kp->ks_instance,
1324                     kp->ks_name, 0);

1326                 /* Save the data necessary to read the kstat info on demand */
1327                 hv_store(tie, "class", 5, newSVpv(kp->ks_class, 0), 0);
1328                 hv_store(tie, "crtime", 6, NEW_HRTIME(kp->ks_crtime), 0);
1329                 kstatinfo.kstat = kp;
1330                 kstatsv = newSVpv((char *)&kstatinfo, sizeof (kstatinfo));
1331                 sv_magic((SV *)tie, kstatsv, '~', 0, 0);
1332                 SvREFCNT_dec(kstatsv);
1333         }
1334         SvREADONLY_on(SvRV(RETVAL));
1335         /* SvREADONLY_on(RETVAL); */
1336 OUTPUT:
1337         RETVAL

1339  #
1340  # Update the perl hash structure so that it is in line with the kernel kstats
1341  # data.  Only kstats athat have previously been accessed are read,
1342  #

1344  # Scalar context: true/false
1345  # Array context: (\@added, \@deleted)
1346 void
1347 update(self)
1348         SV* self;
1349 PREINIT:
1350         MAGIC           *mg;
1351         kstat_ctl_t     *kc;
1352         kstat_t         *kp;
1353         int             ret;
1354         AV              *add, *del;
1355 PPCODE:
1356         /* Find the hidden KstatInfo_t structure */
1357         mg = mg_find(SvRV(self), '~');
1358         PERL_ASSERTMSG(mg != 0, "update: lost ~ magic");
1359         kc = *(kstat_ctl_t **)SvPVX(mg->mg_obj);

1361         /* Update the kstat chain, and return immediately on error. */
1362         if ((ret = kstat_chain_update(kc)) == -1) {
1363                 if (GIMME_V == G_ARRAY) {
1364                         EXTEND(SP, 2);
1365                         PUSHs(sv_newmortal());
1366                         PUSHs(sv_newmortal());
1367                 } else {
1368                         EXTEND(SP, 1);
1369                         PUSHs(sv_2mortal(newSViv(ret)));
1370                 }
1371         }

1373         /* Create the arrays to be returned if in an array context */
1374         if (GIMME_V == G_ARRAY) {
1375                 add = newAV();
1376                 del = newAV();
1377         } else {
1378                 add = 0;
1379                 del = 0;
1380         }
```



```
1382         /*
1383          * If the kstat chain hasn't changed we can just reread any stats
1384          * that have already been read
1385          */
1386         if (ret == 0) {
1387                 if (! apply_to_ties(self, (ATTCb_t)read_kstats, (void *)TRUE)) {
1388                         if (GIMME_V == G_ARRAY) {
1389                                 EXTEND(SP, 2);
1390                                 PUSHs(sv_2mortal(newRV_noinc((SV *)add)));
1391                                 PUSHs(sv_2mortal(newRV_noinc((SV *)del)));
1392                         } else {
1393                                 EXTEND(SP, 1);
1394                                 PUSHs(sv_2mortal(newSViv(-1)));
1395                         }
1396                 }

1398         /*
1399          * Otherwise we have to update the Perl structure so that it is in
1400          * agreement with the new kstat chain.  We do this in such a way as to
1401          * retain all the existing structures, just adding or deleting the
1402          * bare minimum.
1403          */
1404         } else {
1405                 KstatInfo_t     kstatinfo;

1407                 /*
1408                  * Step 1: set the 'invalid' flag on each entry
1409                  */
1410                 apply_to_ties(self, &set_valid, (void *)FALSE);

1412                 /*
1413                  * Step 2: Set the 'valid' flag on all entries still in the
1414                  * kernel kstat chain
1415                  */
1416                 kstatinfo.read          = FALSE;
1417                 kstatinfo.valid         = TRUE;
1418                 kstatinfo.kstat_ctl     = kc;
1419                 for (kp = kc->kc_chain; kp != 0; kp = kp->ks_next) {
1420                         int     new;
1421                         HV      *tie;

1423                         /* Don't bother storing the kstat headers or types */
1424                         if (strncmp(kp->ks_name, "kstat_", 6) == 0) {
1425                                 continue;
1426                         }

1428                         /* Don't bother storing raw stats we don't understand */
1429                         if (kp->ks_type == KSTAT_TYPE_RAW &&
1430                             lookup_raw_kstat_fn(kp->ks_module, kp->ks_name)
1431                             == 0) {
1432 #ifdef REPORT_UNKNOWN
1433                                 (void) printf("Unknown kstat type %s:%d:%s "
1434                                     "- %d of size %d\n", kp->ks_module,
1435                                     kp->ks_instance, kp->ks_name,
1436                                     kp->ks_ndata, kp->ks_data_size);
1437 #endif
1438                                 continue;
1439                         }

1441                         /* Find the tied hash associated with the kstat entry */
1442                         tie = get_tie(self, kp->ks_module, kp->ks_instance,
1443                             kp->ks_name, &new);

1445                         /* If newly created store the associated kstat info */
1446                         if (new) {
```

```
1447                                SV *kstatsv;

1449                                /*
1450                                 * Save the data necessary to read the kstat
1451                                 * info on demand
1452                                 */
1453                                hv_store(tie, "class", 5,
1454                                    newSVpv(kp->ks_class, 0), 0);
1455                                hv_store(tie, "crtime", 6,
1456                                    NEW_HRTIME(kp->ks_crtime), 0);
1457                                kstatinfo.kstat = kp;
1458                                kstatsv = newSVpv((char *)&kstatinfo,
1459                                    sizeof (kstatinfo));
1460                                sv_magic((SV *)tie, kstatsv, '~', 0, 0);
1461                                SvREFCNT_dec(kstatsv);

1463                                /* Save the key on the add list, if required */
1464                                if (GIMME_V == G_ARRAY) {
1465                                        av_push(add, newSVpvf("%s:%d:%s",
1466                                            kp->ks_module, kp->ks_instance,
1467                                            kp->ks_name));
1468                                }

1470                        /* If the stats already exist, just update them */
1471                        } else {
1472                                MAGIC *mg;
1473                                KstatInfo_t *kip;

1475                                /* Find the hidden KstatInfo_t */
1476                                mg = mg_find((SV *)tie, '~');
1477                                PERL_ASSERTMSG(mg != 0, "update: lost ~ magic");
1478                                kip = (KstatInfo_t *)SvPVX(mg->mg_obj);

1480                                /* Mark the tie as valid */
1481                                kip->valid = TRUE;

1483                                /* Re-save the kstat_t pointer.  If the kstat
1484                                 * has been deleted and re-added since the last
1485                                 * update, the address of the kstat structure
1486                                 * will have changed, even though the kstat will
1487                                 * still live at the same place in the perl
1488                                 * hash tree structure.
1489                                 */
1490                                kip->kstat = kp;

1492                                /* Reread the stats, if read previously */
1493                                read_kstats(tie, TRUE);
1494                        }
1495                }

1497                /*
1498                 *Step 3: Delete any entries still marked as 'invalid'
1499                 */
1500                ret = prune_invalid(self, del);

1502        }
1503        if (GIMME_V == G_ARRAY) {
1504                EXTEND(SP, 2);
1505                PUSHs(sv_2mortal(newRV_noinc((SV *)add)));
1506                PUSHs(sv_2mortal(newRV_noinc((SV *)del)));
1507        } else {
1508                EXTEND(SP, 1);
1509                PUSHs(sv_2mortal(newSViv(ret)));
1510        }
```



```
1513  #
1514  # Destructor.  Closes the kstat connection
1515  #

1517 void
1518 DESTROY(self)
1519         SV *self;
1520 PREINIT:
1521         MAGIC           *mg;
1522         kstat_ctl_t     *kc;
1523 CODE:
1524         mg = mg_find(SvRV(self), '~');
1525         PERL_ASSERTMSG(mg != 0, "DESTROY: lost ~ magic");
1526         kc = *(kstat_ctl_t **)SvPVX(mg->mg_obj);
1527         if (kstat_close(kc) != 0) {
1528                 croak(DEBUG_ID ": kstat_close: failed");
1529         }

1531  #
1532  # The following XS methods implement the TIEHASH mechanism used to update the
1533  # kstats hash structure.  These are blessed into a package that isn't
1534  # visible to callers of the Sun::Solaris::Kstat module
1535  #

1537 MODULE = Sun::Solaris::Kstat PACKAGE = Sun::Solaris::Kstat::_Stat
1538 PROTOTYPES: ENABLE

1540  #
1541  # If a value has already been read, return it.  Otherwise read the appropriate
1542  # kstat and then return the value
1543  #

1545 SV*
1546 FETCH(self, key)
1547         SV* self;
1548         SV* key;
1549 PREINIT:
1550         char    *k;
1551         STRLEN  klen;
1552         SV      **value;
1553 CODE:
1554         self = SvRV(self);
1555         k = SvPV(key, klen);
1556         if (strNE(k, "class") && strNE(k, "crtime")) {
1557                 read_kstats((HV *)self, FALSE);
1558         }
1559         value = hv_fetch((HV *)self, k, klen, FALSE);
1560         if (value) {
1561                 RETVAL = *value; SvREFCNT_inc(RETVAL);
1562         } else {
1563                 RETVAL = &PL_sv_undef;
1564         }
1565 OUTPUT:
1566         RETVAL

1568  #
1569  # Save the passed value into the kstat hash.  Read the appropriate kstat first,
1570  # if necessary.  Note that this DOES NOT update the underlying kernel kstat
1571  # structure.
1572  #

1574 SV*
1575 STORE(self, key, value)
1576         SV* self;
1577         SV* key;
1578         SV* value;
```

```
1579 PREINIT:
1580         char    *k;
1581         STRLEN  klen;
1582 CODE:
1583         self = SvRV(self);
1584         k = SvPV(key, klen);
1585         if (strNE(k, "class") && strNE(k, "crtime")) {
1586                 read_kstats((HV *)self, FALSE);
1587         }
1588         SvREFCNT_inc(value);
1589         RETVAL = *(hv_store((HV *)self, k, klen, value, 0));
1590         SvREFCNT_inc(RETVAL);
1591 OUTPUT:
1592         RETVAL

1594  #
1595  # Check for the existence of the passed key.  Read the kstat first if necessary
1596  #

1598 bool
1599 EXISTS(self, key)
1600         SV* self;
1601         SV* key;
1602 PREINIT:
1603         char *k;
1604 CODE:
1605         self = SvRV(self);
1606         k = SvPV(key, PL_na);
1607         if (strNE(k, "class") && strNE(k, "crtime")) {
1608                 read_kstats((HV *)self, FALSE);
1609         }
1610         RETVAL = hv_exists_ent((HV *)self, key, 0);
1611 OUTPUT:
1612         RETVAL


1615  #
1616  # Hash iterator initialisation.  Read the kstats if necessary.
1617  #

1619 SV*
1620 FIRSTKEY(self)
1621         SV* self;
1622 PREINIT:
1623         HE *he;
1624 PPCODE:
1625         self = SvRV(self);
1626         read_kstats((HV *)self, FALSE);
1627         hv_iterinit((HV *)self);
1628         if ((he = hv_iternext((HV *)self))) {
 755         if (he = hv_iternext((HV *)self)) {
1629                 EXTEND(SP, 1);
1630                 PUSHs(hv_iterkeysv(he));
1631         }

1633  #
1634  # Return hash iterator next value.  Read the kstats if necessary.
1635  #

1637 SV*
1638 NEXTKEY(self, lastkey)
1639         SV* self;
1640         SV* lastkey;
1641 PREINIT:
1642         HE *he;
1643 PPCODE:
```



```
1644         self = SvRV(self);
1645         if ((he = hv_iternext((HV *)self))) {
 772         if (he = hv_iternext((HV *)self)) {
1646                 EXTEND(SP, 1);
1647                 PUSHs(hv_iterkeysv(he));
1648         }


1651  #
1652  # Delete the specified hash entry.
1653  #

1655 SV*
1656 DELETE(self, key)
1657         SV *self;
1658         SV *key;
1659 CODE:
1660         self = SvRV(self);
1661         RETVAL = hv_delete_ent((HV *)self, key, 0, 0);
1662         if (RETVAL) {
1663                 SvREFCNT_inc(RETVAL);
1664         } else {
1665                 RETVAL = &PL_sv_undef;
1666         }
1667 OUTPUT:
1668         RETVAL

1670  #
1671  # Clear the entire hash.  This will stop any update() calls rereading this
1672  # kstat until it is accessed again.
1673  #

1675 void
1676 CLEAR(self)
1677         SV* self;
1678 PREINIT:
1679         MAGIC   *mg;
1680         KstatInfo_t *kip;
1681 CODE:
1682         self = SvRV(self);
1683         hv_clear((HV *)self);
1684         mg = mg_find(self, '~');
1685         PERL_ASSERTMSG(mg != 0, "CLEAR: lost ~ magic");
1686         kip = (KstatInfo_t *)SvPVX(mg->mg_obj);
1687         kip->read  = FALSE;
1688         kip->valid = TRUE;
1689         hv_store((HV *)self, "class", 5, newSVpv(kip->kstat->ks_class, 0), 0);
1690         hv_store((HV *)self, "crtime", 6, NEW_HRTIME(kip->kstat->ks_crtime), 0);
```

```
**********************************************************
     904 Fri Feb 21 02:07:13 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Kstat/mapfile-vers
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
#

$mapfile_version 2

SYMBOL_SCOPE {
    global:
        boot_Sun__Solaris__Kstat;
        XS_Sun__Solaris__Kstat___Stat_CLEAR;
        XS_Sun__Solaris__Kstat___Stat_DELETE;
        XS_Sun__Solaris__Kstat___Stat_EXISTS;
        XS_Sun__Solaris__Kstat___Stat_FETCH;
        XS_Sun__Solaris__Kstat___Stat_FIRSTKEY;
        XS_Sun__Solaris__Kstat___Stat_NEXTKEY;
        XS_Sun__Solaris__Kstat___Stat_STORE;
        XS_Sun__Solaris__Kstat_DESTROY;
        XS_Sun__Solaris__Kstat_new;
        XS_Sun__Solaris__Kstat_update;
    local:
        *;
};
#endif /* ! codereview */
```

```
**********************************************************
    6928 Fri Feb 21 02:07:13 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Lgrp/Lgrp.pm
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
#
# 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 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# Copyright (c) 2014 Racktop Systems.
#endif /* ! codereview */
#

#
# Lgrp.pm provides procedural and object-oriented interface to the Solaris
# liblgrp(3LIB) library.
#


require 5.0010;
require 5.8.4;
use strict;
use warnings;
use Carp;

package Sun::Solaris::Lgrp;

our $VERSION = '1.1';
use XSLoader;
XSLoader::load(__PACKAGE__, $VERSION);

require Exporter;

our @ISA = qw(Exporter);

our (@EXPORT_OK, %EXPORT_TAGS);

# Things to export
my @lgrp_constants = qw(LGRP_AFF_NONE LGRP_AFF_STRONG LGRP_AFF_WEAK
                        LGRP_CONTENT_DIRECT LGRP_CONTENT_HIERARCHY
                        LGRP_MEM_SZ_FREE LGRP_MEM_SZ_INSTALLED LGRP_VER_CURRENT
                        LGRP_VER_NONE LGRP_VIEW_CALLER
                        LGRP_VIEW_OS LGRP_NONE
```



```
                        LGRP_RSRC_CPU LGRP_RSRC_MEM
                        LGRP_CONTENT_ALL LGRP_LAT_CPU_TO_MEM
);

my @proc_constants = qw(P_PID P_LWPID P_MYID);

my @constants = (@lgrp_constants, @proc_constants);

my @functions = qw(lgrp_affinity_get lgrp_affinity_set
                   lgrp_children lgrp_cookie_stale lgrp_cpus lgrp_fini
                   lgrp_home lgrp_init lgrp_latency lgrp_latency_cookie
                   lgrp_mem_size lgrp_nlgrps lgrp_parents
                   lgrp_root lgrp_version lgrp_view lgrp_resources
                   lgrp_isleaf lgrp_lgrps lgrp_leaves);

my @all = (@constants, @functions);

# Define symbolic names for various subsets of export lists
%EXPORT_TAGS = ('CONSTANTS' => \@constants,
                'LGRP_CONSTANTS' => \@lgrp_constants,
                'PROC_CONSTANTS' => \@proc_constants,
                'FUNCTIONS' => \@functions,
                'ALL' => \@all);

# Define things that are ok ot export.
@EXPORT_OK = ( @{ $EXPORT_TAGS{'ALL'} } );

#
# _usage(): print error message and terminate the program.
#
sub _usage
{
        my $msg = shift;
        Carp::croak "Usage: Sun::Solaris::Lgrp::$msg";
}
______unchanged_portion_omitted_
```

```
**********************************************************
    1172 Fri Feb 21 02:07:14 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Lgrp/mapfile-vers
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
#

$mapfile_version 2

SYMBOL_SCOPE {
    global:
        boot_Sun__Solaris__Lgrp;
        XS_Sun__Solaris__Lgrp_lgrp_affinity_get;
        XS_Sun__Solaris__Lgrp_lgrp_affinity_set;
        XS_Sun__Solaris__Lgrp_lgrp_children;
        XS_Sun__Solaris__Lgrp_lgrp_cookie_stale;
        XS_Sun__Solaris__Lgrp_lgrp_cpus;
        XS_Sun__Solaris__Lgrp_lgrp_fini;
        XS_Sun__Solaris__Lgrp_lgrp_home;
        XS_Sun__Solaris__Lgrp_lgrp_init;
        XS_Sun__Solaris__Lgrp_lgrp_latency;
        XS_Sun__Solaris__Lgrp_lgrp_latency_cookie;
        XS_Sun__Solaris__Lgrp_lgrp_mem_size;
        XS_Sun__Solaris__Lgrp_lgrp_nlgrps;
        XS_Sun__Solaris__Lgrp_lgrp_parents;
        XS_Sun__Solaris__Lgrp_lgrp_resources;
        XS_Sun__Solaris__Lgrp_lgrp_root;
        XS_Sun__Solaris__Lgrp_lgrp_version;
        XS_Sun__Solaris__Lgrp_lgrp_view;
    local:
        *;
};
#endif /* ! codereview */
```

**********
1088 Fri Feb 21 02:07:15 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Project/mapfile-vers
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
#

$mapfile_version 2

SYMBOL_SCOPE {
    global:
        boot_Sun__Solaris__Project;
        XS_Sun__Solaris__Project_activeprojects;
        XS_Sun__Solaris__Project_endprojent;
        XS_Sun__Solaris__Project_fgetprojent;
        XS_Sun__Solaris__Project_getdefaultproj;
        XS_Sun__Solaris__Project_getprojbyid;
        XS_Sun__Solaris__Project_getprojbyname;
        XS_Sun__Solaris__Project_getprojent;
        XS_Sun__Solaris__Project_getprojid;
        XS_Sun__Solaris__Project_getprojidbyname;
        XS_Sun__Solaris__Project_inproj;
        XS_Sun__Solaris__Project_pool_exists;
        XS_Sun__Solaris__Project_rctl_get_info;
        XS_Sun__Solaris__Project_setproject;
        XS_Sun__Solaris__Project_setprojent;
    local:
        *;
};
#endif /* ! codereview */
```

```
**********************************************************
   41471 Fri Feb 21 02:07:15 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Project/Project.pm
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
#
# Copyright (c) 1999, 2008, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2014 Racktop Systems.
#endif /* ! codereview */
#

#
# Project.pm provides the bootstrap for the Sun::Solaris::Project module, and
# also functions for reading, validating and writing out project(4) format
# files.
#
################################################################################
require 5.0010;
require 5.8.4;

use strict;
use warnings;
use locale;
use Errno;
use Fcntl;
use File::Basename;
use POSIX qw(locale_h limits_h);

package Sun::Solaris::Project;

our $VERSION = '1.9';

use XSLoader;
XSLoader::load(__PACKAGE__, $VERSION);

our (@EXPORT_OK, %EXPORT_TAGS);
my @constants = qw(MAXPROJID PROJNAME_MAX PROJF_PATH PROJECT_BUFSZ
    SETPROJ_ERR_TASK SETPROJ_ERR_POOL);
my @syscalls = qw(getprojid);
my @libcalls = qw(setproject activeprojects getprojent setprojent endprojent
    getprojbyname getprojbyid getdefaultproj fgetprojent inproj
    getprojidbyname);
my @private = qw(projf_read projf_write projf_validate projent_parse
                 projent_parse_name projent_validate_unique_name
                 projent_parse_projid projent_validate_unique_id
                 projent_parse_comment
                 projent_parse_users
                 projent_parse_groups
                 projent_parse_attributes
                 projent_validate projent_validate_projid
                 projent_values_equal projent_values2string);

@EXPORT_OK = (@constants, @syscalls, @libcalls, @private);
%EXPORT_TAGS = (CONSTANTS => \@constants, SYSCALLS => \@syscalls,
    LIBCALLS => \@libcalls, PRIVATE => \@private, ALL => \@EXPORT_OK);

use base qw(Exporter);
use Sun::Solaris::Utils qw(gettext);

#
# Set up default rules for validating rctls.
# These rules are not global-flag specific, but instead
# are the total set of allowable values on all rctls.
```



```
#
use Config;
our $MaxNum = &RCTL_MAX_VALUE;
our %RctlRules;

my %rules;
our %SigNo;
my $j;
my $name;
foreach $name (split(' ', $Config{sig_name})) {
        $SigNo{$name} = $j;
        $j++;
}
______unchanged_portion_omitted_
```

```
**********************************************************
    8627 Fri Feb 21 02:07:15 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Project/Project.xs
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
/*
 * Copyright (c) 1999, 2007, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2014 Racktop Systems.
#endif /* ! codereview */
 */
/*
 * Project.xs contains XS wrappers for the project database maniplulation
 * functions as provided by libproject and described in getprojent(3EXACCT).
 */

/* Solaris includes. */
#include <zone.h>
#include <project.h>
#include <pool.h>
#include <sys/pool_impl.h>
#include <rctl.h>
#include <stdio.h>

/* Perl includes. */
#include "EXTERN.h"
#include "perl.h"
#include "XSUB.h"

/*
 * Convert and save a struct project on the perl XS return stack.
 * In a void context it returns nothing, in a scalar context it returns just
 * the name of the project and in a list context it returns a 6-element list
 * consisting of (name, projid, comment, users, groups, attr), where users and
 * groups are references to arrays containing the appropriate lists.
 */
static int
pushret_project(const struct project *proj)
{
        char    **cp;
        AV      *ary;

        dSP;
        if (GIMME_V == G_SCALAR) {
                EXTEND(SP, 1);
                PUSHs(sv_2mortal(newSVpv(proj->pj_name, 0)));
                PUTBACK;
                return (1);
        } else if (GIMME_V == G_ARRAY) {
                EXTEND(SP, 6);
                PUSHs(sv_2mortal(newSVpv(proj->pj_name, 0)));
                PUSHs(sv_2mortal(newSViv(proj->pj_projid)));
                PUSHs(sv_2mortal(newSVpv(proj->pj_comment, 0)));
                ary = newAV();
                for (cp = proj->pj_users; *cp != NULL; cp++) {
                        av_push(ary, newSVpv(*cp, 0));
                }
                PUSHs(sv_2mortal(newRV_noinc((SV *)ary)));
                ary = newAV();
                for (cp = proj->pj_groups; *cp != NULL; cp++) {
                        av_push(ary, newSVpv(*cp, 0));
                }
                PUSHs(sv_2mortal(newRV_noinc((SV *)ary)));
                PUSHs(sv_2mortal(newSVpv(proj->pj_attr, 0)));
```



```
                PUTBACK;
                return (6);
        } else {
                return (0);
        }
}

static int
pwalk_cb(const projid_t project, void *walk_data)
{
        int *nitemsp;

        dSP;
        nitemsp = (int *) walk_data;
        EXTEND(SP, 1);
        PUSHs(sv_2mortal(newSViv(project)));
        (*nitemsp)++;
        PUTBACK;
        return (0);
}

/*
 * The XS code exported to perl is below here.  Note that the XS preprocessor
 * has its own commenting syntax, so all comments from this point on are in
 * that form.  Note also that the PUTBACK; lines are necessary to synchronise
 * the local and global views of the perl stack before calling pushret_project,
 * as the code generated by the perl XS compiler twiddles with the stack on
 * entry to an XSUB.
 */

MODULE = Sun::Solaris::Project PACKAGE = Sun::Solaris::Project
PROTOTYPES: ENABLE

 #
 # Define any constants that need to be exported.  By doing it this way we can
 # avoid the overhead of using the DynaLoader package, and in addition constants
 # defined using this mechanism are eligible for inlining by the perl
 # interpreter at compile time.
 #
BOOT:
        {
        HV *stash;
        char buf[128];
        stash = gv_stashpv("Sun::Solaris::Project", TRUE);
        newCONSTSUB(stash, "MAXPROJID", newSViv(MAXPROJID));
        newCONSTSUB(stash, "PROJNAME_MAX", newSViv(PROJNAME_MAX));
        newCONSTSUB(stash, "PROJF_PATH",
            newSVpv(PROJF_PATH, sizeof (PROJF_PATH) - 1));
        newCONSTSUB(stash, "PROJECT_BUFSZ", newSViv(PROJECT_BUFSZ));
        newCONSTSUB(stash, "SETPROJ_ERR_TASK", newSViv(SETPROJ_ERR_TASK));
        newCONSTSUB(stash, "SETPROJ_ERR_POOL", newSViv(SETPROJ_ERR_POOL));
        newCONSTSUB(stash, "RCTL_GLOBAL_NOBASIC",
                newSViv(RCTL_GLOBAL_NOBASIC));
        newCONSTSUB(stash, "RCTL_GLOBAL_LOWERABLE",
                newSViv(RCTL_GLOBAL_LOWERABLE));
        newCONSTSUB(stash, "RCTL_GLOBAL_DENY_ALWAYS",
                newSViv(RCTL_GLOBAL_DENY_ALWAYS));
        newCONSTSUB(stash, "RCTL_GLOBAL_DENY_NEVER",
                newSViv(RCTL_GLOBAL_DENY_NEVER));
        newCONSTSUB(stash, "RCTL_GLOBAL_FILE_SIZE",
                newSViv(RCTL_GLOBAL_FILE_SIZE));
        newCONSTSUB(stash, "RCTL_GLOBAL_CPU_TIME",
                newSViv(RCTL_GLOBAL_CPU_TIME));
        newCONSTSUB(stash, "RCTL_GLOBAL_SIGNAL_NEVER",
                newSViv(RCTL_GLOBAL_SIGNAL_NEVER));
        newCONSTSUB(stash, "RCTL_GLOBAL_INFINITE",
```

```
125                 newSViv(RCTL_GLOBAL_INFINITE));
126         newCONSTSUB(stash, "RCTL_GLOBAL_UNOBSERVABLE",
127                 newSViv(RCTL_GLOBAL_UNOBSERVABLE));
128         newCONSTSUB(stash, "RCTL_GLOBAL_BYTES",
129                 newSViv(RCTL_GLOBAL_BYTES));
130         newCONSTSUB(stash, "RCTL_GLOBAL_SECONDS",
131                 newSViv(RCTL_GLOBAL_SECONDS));
132         newCONSTSUB(stash, "RCTL_GLOBAL_COUNT",
133                 newSViv(RCTL_GLOBAL_COUNT));
134         sprintf(buf, "%llu", UINT64_MAX);
135         newCONSTSUB(stash, "RCTL_MAX_VALUE",
136                 newSVpv(buf, strlen(buf)));
137         }

139 projid_t
140 getprojid()

142 int
143 setproject(name, user_name, flags)
144         const char      *name;
145         const char      *user_name
146         uint_t          flags

148 void
149 activeprojects()
150 PREINIT:
151         int     nitems;
152 PPCODE:
153         PUTBACK;
154         nitems = 0;
155         project_walk(&pwalk_cb, (void*)&nitems);
156         XSRETURN(nitems);

158 void
159 getprojent()
160 PREINIT:
161         struct project  proj, *projp;
162         char            buf[PROJECT_BUFSZ];
163 PPCODE:
164         PUTBACK;
165         if ((projp = getprojent(&proj, buf, sizeof (buf)))) {
  3         if (projp = getprojent(&proj, buf, sizeof (buf))) {
166                 XSRETURN(pushret_project(projp));
167         } else {
168                 XSRETURN_EMPTY;
169         }

171 void
172 setprojent()

174 void
175 endprojent()

177 void
178 getprojbyname(name)
179         char    *name
180 PREINIT:
181         struct project  proj, *projp;
182         char            buf[PROJECT_BUFSZ];
183 PPCODE:
184         PUTBACK;
185         if ((projp = getprojbyname(name, &proj, buf, sizeof (buf)))) {
 23         if (projp = getprojbyname(name, &proj, buf, sizeof (buf))) {
186                 XSRETURN(pushret_project(projp));
187         } else {
188                 XSRETURN_EMPTY;
```



```
189         }

191 void
192 getprojbyid(id)
193         projid_t        id
194 PREINIT:
195         struct project  proj, *projp;
196         char            buf[PROJECT_BUFSZ];
197 PPCODE:
198         PUTBACK;
199         if ((projp = getprojbyid(id, &proj, buf, sizeof (buf)))) {
 37         if (projp = getprojbyid(id, &proj, buf, sizeof (buf))) {
200                 XSRETURN(pushret_project(projp));
201         } else {
202                 XSRETURN_EMPTY;
203         }

205 void
206 getdefaultproj(user)
207         char    *user
208 PREINIT:
209         struct project  proj, *projp;
210         char            buf[PROJECT_BUFSZ];
211 PPCODE:
212         PUTBACK;
213         if ((projp = getdefaultproj(user, &proj, buf, sizeof (buf)))) {
 51         if (projp = getdefaultproj(user, &proj, buf, sizeof (buf))) {
214                 XSRETURN(pushret_project(projp));
215         } else {
216                 XSRETURN_EMPTY;
217         }

219 void
220 fgetprojent(fh)
221         FILE    *fh
222 PREINIT:
223         struct project  proj, *projp;
224         char            buf[PROJECT_BUFSZ];
225 PPCODE:
226         PUTBACK;
227         if ((projp = fgetprojent(fh, &proj, buf, sizeof (buf)))) {
 65         if (projp = fgetprojent(fh, &proj, buf, sizeof (buf))) {
228                 XSRETURN(pushret_project(projp));
229         } else {
230                 XSRETURN_EMPTY;
231         }

233 bool
234 inproj(user, proj)
235         char    *user
236         char    *proj
237 PREINIT:
238         char    buf[PROJECT_BUFSZ];
239 CODE:
240         RETVAL = inproj(user, proj, buf, sizeof (buf));
241 OUTPUT:
242         RETVAL
243 #endif /* ! codereview */


246 int
247 getprojidbyname(proj)
248         char    *proj
249 PREINIT:
250         int     id;
251 PPCODE:
```

```
252         if ((id = getprojidbyname(proj)) == -1) {
253                 XSRETURN_UNDEF;
254         } else {
255                 XSRETURN_IV(id);
256         }


259 # rctl_get_info(name)
260 #
261 # For the given rctl name, returns the list
262 # ($max, $flags), where $max is the integer value
263 # of the system rctl, and $flags are the rctl's
264 # global flags, as returned by rctlblk_get_global_flags
265 #
266 # This function is private to Project.pm
267 void
268 rctl_get_info(name)
269         char    *name
270 PREINIT:
271         rctlblk_t *blk1 = NULL;
272         rctlblk_t *blk2 = NULL;
273         rctlblk_t *tmp = NULL;
274         rctl_priv_t priv;
275         rctl_qty_t value;
276         int flags = 0;
 79         int flags;
277         int ret;
278         int err = 0;
279         char string[24];        /* 24 will always hold a uint64_t */
280 PPCODE:
281         Newc(0, blk1, rctlblk_size(), char, rctlblk_t);
282         if (blk1 == NULL) {
283                 err = 1;
284                 goto out;
285         }
286         Newc(1, blk2, rctlblk_size(), char, rctlblk_t);
287         if (blk2 == NULL) {
288                 err = 1;
289                 goto out;
290         }
291         ret = getrctl(name, NULL, blk1, RCTL_FIRST);
292         if (ret != 0) {
293                 err = 1;
294                 goto out;
295         }
296         priv = rctlblk_get_privilege(blk1);
297         while (priv != RCPRIV_SYSTEM) {
298                 tmp = blk2;
299                 blk2 = blk1;
300                 blk1 = tmp;
301                 ret = getrctl(name, blk2, blk1, RCTL_NEXT);
302                 if (ret != 0) {
303                         err = 1;
304                         goto out;
305                 }
306                 priv = rctlblk_get_privilege(blk1);
307         }
308         value = rctlblk_get_value(blk1);
309         flags = rctlblk_get_global_flags(blk1);
310         ret = sprintf(string, "%llu", value);
311         if (ret <= 0) {
312                 err = 1;
313         }
314         out:
315         if (blk1)
316                 Safefree(blk1);
```



```
317         if (blk2)
318                 Safefree(blk2);
319         if (err)
320                 XSRETURN(0);

322         XPUSHs(sv_2mortal(newSVpv(string, 0)));
323         XPUSHs(sv_2mortal(newSViv(flags)));
324         XSRETURN(2);

326 #
327 # pool_exists(name)
328 #
329 # Returns 0 a pool with the given name exists on the current system.
330 # Returns 1 if pools are disabled or the pool does not exist
331 #
332 # Used internally by project.pm to validate the project.pool attribute
333 #
334 # This function is private to Project.pm
335 void
336 pool_exists(name)
337         char    *name
338 PREINIT:
339         pool_conf_t *conf;
340         pool_t *pool;
341         pool_status_t status;
342         int fd;
343 PPCODE:

345         /*
346          * Determine if pools are enabled using /dev/pool directly, as
347          * libpool may not be present.
348          */
349         if (getzoneid() != GLOBAL_ZONEID) {
350                 XSRETURN_IV(1);
351         }
352         if ((fd = open("/dev/pool", O_RDONLY)) < 0) {
353                 XSRETURN_IV(1);
354         }
355         if (ioctl(fd, POOL_STATUSQ, &status) < 0) {
356                 (void) close(fd);
357                 XSRETURN_IV(1);
358         }
359         close(fd);
360         if (status.ps_io_state != 1) {
361                 XSRETURN_IV(1);
362         }

364         /*
365          * If pools are enabled, assume libpool is present.
366          */
367         conf = pool_conf_alloc();
368         if (conf == NULL) {
369                 XSRETURN_IV(1);
370         }
371         if (pool_conf_open(conf, pool_dynamic_location(), PO_RDONLY)) {
372                 pool_conf_free(conf);
373                 XSRETURN_IV(1);
374         }
375         pool = pool_get_pool(conf, name);
376         if (pool == NULL) {
377                 pool_conf_close(conf);
378                 pool_conf_free(conf);
379                 XSRETURN_IV(1);
380         }
381         pool_conf_close(conf);
382         pool_conf_free(conf);
```

```
 383         XSRETURN_IV(0);
```

```
**********************************************************
     696 Fri Feb 21 02:07:16 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Task/Makefile
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
#

MODULE = Task

include $(SRC)/cmd/perl/Makefile.perl

XSUBPPFLAGS = -typemap typemap

MAPFILES = mapfile-vers

include $(SRC)/cmd/perl/Makefile.targ

all: $(PERLEXT) $(PERLMOD)

install: $(ROOTPERLEXT) $(ROOTPERLMOD)

clean clobber:
        $(RM) -r $(MACH)
#endif /* ! codereview */
```

```
**********************************************************
     632 Fri Feb 21 02:07:16 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Task/Task.pm
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
   1 #
   2 # Copyright (c) 2002, 2008, Oracle and/or its affiliates. All rights reserved.
   3 # Copyright (c) 2014 Racktop Systems.
   4 #endif /* ! codereview */
   5 #

   7 #
   8 # Task.pm provides the bootstrap for the Sun::Solaris::Task module.
   9 #

  11 require 5.0010;
   3 require 5.8.4;
  12 use strict;
  13 use warnings;

  15 package Sun::Solaris::Task;

  17 our $VERSION = '1.4';
  18 use XSLoader;
  19 XSLoader::load(__PACKAGE__, $VERSION);

  21 our (@EXPORT_OK, %EXPORT_TAGS);
  22 my @constants = qw(TASK_NORMAL TASK_FINAL TASK_PROJ_PURGE);
  23 my @syscalls = qw(settaskid gettaskid);
  24 @EXPORT_OK = (@constants, @syscalls);
  25 %EXPORT_TAGS = (CONSTANTS => \@constants, SYSCALLS => \@syscalls,
  26     ALL => \@EXPORT_OK);

  28 use base qw(Exporter);

  30 1;
```

```
**********************************************************
     604 Fri Feb 21 02:07:16 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Task/mapfile-vers
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
#

$mapfile_version 2

SYMBOL_SCOPE {
    global:
        boot_Sun__Solaris__Task;
        XS_Sun__Solaris__Task_gettaskid;
        XS_Sun__Solaris__Task_settaskid;
    local:
        *;
};
#endif /* ! codereview */
```

```
**********************************************************
     747 Fri Feb 21 02:07:17 2014
new/usr/src/cmd/perl/contrib/Sun/Solaris/Utils/mapfile-vers
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
#

$mapfile_version 2

SYMBOL_SCOPE {
    global:
        boot_Sun__Solaris__Utils;
        XS_Sun__Solaris__Utils_bindtextdomain;
        XS_Sun__Solaris__Utils_dcgettext;
        XS_Sun__Solaris__Utils_dgettext;
        XS_Sun__Solaris__Utils_gettext;
        XS_Sun__Solaris__Utils_gmatch;
        XS_Sun__Solaris__Utils_textdomain;
    local:
        *;
};
#endif /* ! codereview */
```

```
**********************************************************
     756 Fri Feb 21 02:07:18 2014
new/usr/src/man/man3perl/Makefile
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet
# at http://www.illumos.org/license/CDDL.
#

#
# Copyright 2011, Richard Lowe
# Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
# Copyright (c) 2014 Racktop Systems.
#endif /* ! codereview */
#

include         $(SRC)/Makefile.master

MANSECT=        3perl

MANFILES=       Kstat.3perl                     \
MANFILES=       Exacct.3perl                    \
                Exacct\:\:Catalog.3perl         \
                Exacct\:\:File.3perl            \
                Exacct\:\:Object.3perl          \
                Exacct\:\:Object\:\:Group.3perl \
                Exacct\:\:Object\:\:Item.3perl  \
                Kstat.3perl                     \
                Lgrp.3perl                      \
                Privilege.3perl                 \
                Project.3perl                   \
                Task.3perl
                Task.3perl                      \
                Ucred.3perl

.KEEP_STATE:

include         $(SRC)/man/Makefile.man

install: $(ROOTMANFILES) $(ROOTMANLINKS)
#
# When KEEP_STATE is in effect and a target has a colon in the name (like the
# Exacct::* pages above, dmake will write them to the state file unescaped,
# creating a file which then cannot be reparsed, breaking any build other than
# the first in this directory:
#
# See CR 6987108 make will write un-escaped :'s to .make.state, break itself
#
# As a workaround, install the files manually in a FRC target.
CMD= $(INS) -s -m $(FILEMODE) -f $(ROOTMAN)/man$(MANSECT)
install: FRC
        @for file in $(MANFILES); do                                    \
                if [[ $$file -nt $(ROOTMAN)/man$(MANSECT)/$$file ]]; then \
                        $(ECHO) $(CMD) $$file;                          \
                        $(RM) $(ROOTMAN)/man$(MANSECT)/$$file;          \
                        $(CMD) $$file;                                  \
```



```
                fi                                                      \
        done;
```

```
**********************************************************
    3926 Fri Feb 21 02:07:18 2014
new/usr/src/pkg/manifests/runtime-perl-510-module-sun-solaris.mf
3900 illumos will not build against gcc compiled perl
4723 Remove unused perl extensions
Reviewed by: Keith Wesolowski <keith.wesolowski@joyent.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
**********************************************************
#
# 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 (c) 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2014 Racktop Systems.
#endif /* ! codereview */
#

$(i386_ONLY)<transform file dir path=.*PLAT.* -> edit path PLAT i86pc>
$(sparc_ONLY)<transform file dir path=.*PLAT.* -> edit path PLAT sun4>

<transform file path=.*\.(pm|bs) -> default mode 0444>
<transform file path=.*\.so -> default mode 0555>
set name=pkg.fmri \
    value=pkg:/runtime/perl-510/module/sun-solaris@0.5.11,$(PKGVERS_BUILTON)-$(P
set name=pkg.summary value="Perl 5.10.0 Sun::Solaris Modules"
set name=info.classification \
    value=org.opensolaris.category.2008:Development/Perl
set name=variant.arch value=$(ARCH)
dir path=usr group=sys
dir path=usr/bin
dir path=usr/perl5
dir path=usr/perl5/5.10.0
dir path=usr/perl5/5.10.0/bin
dir path=usr/perl5/5.10.0/lib
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Exacct
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/BSM
dir \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/BSM/_BSMparse
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct
dir \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct/Catalog
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct/File
```



```
dir \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct/Object
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Intrs
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Kstat
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Lgrp
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/PerlGcc
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Pg
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Privilege
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Project
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Task
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Ucred
dir path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Utils
dir path=usr/perl5/5.10.0/lib/Sun
dir path=usr/perl5/5.10.0/lib/Sun/Solaris
dir path=usr/perl5/5.10.0/lib/Sun/Solaris/BSM
dir path=usr/perl5/5.10.0/lib/Sun/Solaris/PerlGcc
dir path=usr/share/man
dir path=usr/share/man/man3perl
file path=usr/perl5/5.10.0/bin/perlgcc mode=0555
file path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Exacct.pm
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Exacct/Catalog.pm
file path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Exacct/File.pm
file path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Exacct/Object.pm
file path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Intrs.pm
file path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Kstat.pm
file path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Lgrp.pm
file path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Privilege.pm
file path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Project.pm
file path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Task.pm
file path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Ucred.pm
file path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/Sun/Solaris/Utils.pm
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct/Catalog
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct/Catalog
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct/Exacct.
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct/Exacct.
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct/File/Fi
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct/File/Fi
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct/Object/
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Exacct/Object/
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Intrs/Intrs.bs
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Intrs/Intrs.so
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Kstat/Kstat.bs
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Kstat/Kstat.so
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Lgrp/Lgrp.bs
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Lgrp/Lgrp.so
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Privilege/Priv
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Privilege/Priv
file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Project/Projec
```

```
*file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Project/Projec
file \
*   path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Task/Task.bs
*file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Task/Task.so
file \
*   path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Ucred/Ucred.bs
*file \
*   path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Ucred/Ucred.so
*file \
*   path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Utils/Utils.bs
*file \
    path=usr/perl5/5.10.0/lib/PLAT-solaris-64int/auto/Sun/Solaris/Utils/Utils.so
file path=usr/perl5/5.10.0/lib/Sun/Solaris/BSM/_BSMparse.pm
*file path=usr/perl5/5.10.0/lib/Sun/Solaris/PerlGcc/Config.pm
file path=usr/perl5/5.10.0/lib/Sun/Solaris/Pg.pm
*file path=usr/share/man/man3perl/Exacct.3perl
*file path=usr/share/man/man3perl/Exacct::Catalog.3perl
*file path=usr/share/man/man3perl/Exacct::File.3perl
*file path=usr/share/man/man3perl/Exacct::Object.3perl
*file path=usr/share/man/man3perl/Exacct::Object::Group.3perl
*file path=usr/share/man/man3perl/Exacct::Object::Item.3perl
file path=usr/share/man/man3perl/Kstat.3perl
file path=usr/share/man/man3perl/Lgrp.3perl
*file path=usr/share/man/man3perl/Privilege.3perl
file path=usr/share/man/man3perl/Project.3perl
file path=usr/share/man/man3perl/Task.3perl
*file path=usr/share/man/man3perl/Ucred.3perl
license cr_Sun license=cr_Sun
license usr/src/cmd/perl/THIRDPARTYLICENSE \
    license=usr/src/cmd/perl/THIRDPARTYLICENSE
depend fmri=runtime/perl-510/extra type=require
```