```
**********************************************************
    2790 Wed Apr  9 02:10:57 2014
new/usr/src/head/uuid/uuid.h
4118 libuuid should provide uuid_unparse_{upper,lower} functions
Reviewed by: Serghei Samsi <sscdvp@gmail.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright 2014 Andrew Stormont.
 */

#endif /* ! codereview */
#ifndef _UUID_H
#define _UUID_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#ifdef  __cplusplus
extern "C" {
#endif

/*
 * The copyright in this file is taken from the original Leach & Salz
 * UUID specification, from which this implementation is derived.
 */

/*
 * Copyright (c) 1990- 1993, 1996 Open Software Foundation, Inc.
 * Copyright (c) 1989 by Hewlett-Packard Company, Palo Alto, Ca. &
 * Digital Equipment Corporation, Maynard, Mass.  Copyright (c) 1998
 * Microsoft.  To anyone who acknowledges that this file is provided
 * "AS IS" without any express or implied warranty: permission to use,
 * copy, modify, and distribute this file for any purpose is hereby
 * granted without fee, provided that the above copyright notices and
 * this notice appears in all source code copies, and that none of the
 * names of Open Software Foundation, Inc., Hewlett-Packard Company,
 * or Digital Equipment Corporation be used in advertising or
 * publicity pertaining to distribution of the software without
```



```
 * specific, written prior permission.  Neither Open Software
 * Foundation, Inc., Hewlett-Packard Company, Microsoft, nor Digital
 * Equipment Corporation makes any representations about the
 * suitability of this software for any purpose.
 */

#include <sys/types.h>
#include <sys/uuid.h>

extern void uuid_generate(uuid_t);
extern void uuid_generate_random(uuid_t);
extern void uuid_generate_time(uuid_t);
extern void uuid_copy(uuid_t, uuid_t);
extern void uuid_clear(uuid_t);
extern void uuid_unparse(uuid_t, char *);
extern void uuid_unparse_lower(uuid_t, char *);
extern void uuid_unparse_upper(uuid_t, char *);
#endif /* ! codereview */
extern int uuid_compare(uuid_t, uuid_t);
extern int uuid_is_null(uuid_t);
extern int uuid_parse(char *, uuid_t);
extern time_t uuid_time(uuid_t, struct timeval *);

#ifdef __cplusplus
}
#endif

#endif /* _UUID_H */
```

```
**********************************************************
    1660 Wed Apr  9 02:10:57 2014
new/usr/src/lib/libuuid/common/mapfile-vers
4118 libuuid should provide uuid_unparse_{upper,lower} functions
Reviewed by: Serghei Samsi <sscdvp@gmail.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
#
# 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) 2006, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
#

#
# MAPFILE HEADER START
#
# WARNING:  STOP NOW.  DO NOT MODIFY THIS FILE.
# Object versioning must comply with the rules detailed in
#
#       usr/src/lib/README.mapfiles
#
# You should not be making modifications here until you've read the most current
# copy of that file. If you need help, contact a gatekeeper for guidance.
#
# MAPFILE HEADER END
#

$mapfile_version 2

SYMBOL_VERSION ILLUMOS_0.1 {
    global:
        uuid_unparse_lower;
        uuid_unparse_upper;
} SUNW_1.1;

#endif /* ! codereview */
SYMBOL_VERSION SUNW_1.1 {
    global:
        uuid_clear;
        uuid_compare;
        uuid_copy;
        uuid_generate;
        uuid_generate_random;
        uuid_generate_time;
```



```
        uuid_is_null;
        uuid_parse;
        uuid_time;
        uuid_unparse;
};

SYMBOL_VERSION SUNWprivate_1.1 {
    global:
        SUNWprivate_1.1;
    local:
        *;
};
```

```
**********************************************************
   17134 Wed Apr  9 02:10:57 2014
new/usr/src/lib/libuuid/common/uuid.c
4118 libuuid should provide uuid_unparse_{upper,lower} functions
Reviewed by: Serghei Samsi <sscdvp@gmail.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
/*
 * 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 2012 Milan Jurik. All rights reserved.
 * Copyright 2013 Joyent, Inc. All rights reserved.
 * Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
 */

/*
 * The copyright in this file is taken from the original Leach & Salz
 * UUID specification, from which this implementation is derived.
 */

/*
 * Copyright (c) 1990- 1993, 1996 Open Software Foundation, Inc.
 * Copyright (c) 1989 by Hewlett-Packard Company, Palo Alto, Ca. &
 * Digital Equipment Corporation, Maynard, Mass.  Copyright (c) 1998
 * Microsoft.  To anyone who acknowledges that this file is provided
 * "AS IS" without any express or implied warranty: permission to use,
 * copy, modify, and distribute this file for any purpose is hereby
 * granted without fee, provided that the above copyright notices and
 * this notice appears in all source code copies, and that none of the
 * names of Open Software Foundation, Inc., Hewlett-Packard Company,
 * or Digital Equipment Corporation be used in advertising or
 * publicity pertaining to distribution of the software without
 * specific, written prior permission.  Neither Open Software
 * Foundation, Inc., Hewlett-Packard Company, Microsoft, nor Digital
 * Equipment Corporation makes any representations about the
 * suitability of this software for any purpose.
 */

/*
 * This module is the workhorse for generating abstract
 * UUIDs.  It delegates system-specific tasks (such
 * as obtaining the node identifier or system time)
 * to the sysdep module.
```



```
 */

#include <ctype.h>
#include <sys/param.h>
#include <sys/stat.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <fcntl.h>
#include <unistd.h>
#include <synch.h>
#include <sys/mman.h>
#include "uuid_misc.h"

shared_buffer_t         *data;

static  uuid_node_t     node_id_cache;
static  int             node_init;
static  int             file_type;
static  int             fd;

/*
 * The urandmtx mutex prevents multiple opens of /dev/urandom and protects the
 * cache.
 */
#define RCACHE_SIZE     65535
static  mutex_t         urandmtx;
static  int             fd_urand = -1;
static  char            rcache[RCACHE_SIZE];
static  char            *rcachep = rcache;

/*
 * misc routines
 */
uint16_t                get_random(void);
void                    get_current_time(uuid_time_t *);

void                    struct_to_string(uuid_t, struct uuid *);
void                    string_to_struct(struct uuid *, uuid_t);
int                     get_ethernet_address(uuid_node_t *);

/*
 * local functions
 */
static  int             map_state();
static  void            format_uuid(struct uuid *, uint16_t, uuid_time_t,
    uuid_node_t);
static  void            fill_random_bytes(uchar_t *, int);
static  int             uuid_create(struct uuid *);
static  void            gen_ethernet_address(uuid_node_t *);
static  void            revalidate_data(uuid_node_t *);

/*
 * Generates a uuid based on version 1 format.
 * Returns 0 on success and -1 on failure.
 */
static int
uuid_create(struct uuid *uuid)
{
        uuid_time_t     timestamp;
        uuid_node_t     system_node;
        int             ret, non_unique = 0;

        /*
         * Get the system MAC address and/or cache it
```

```
124          */
125         if (node_init) {
126                 bcopy(&node_id_cache, &system_node, sizeof (uuid_node_t));
127         } else {
128                 gen_ethernet_address(&system_node);
129                 bcopy(&system_node, &node_id_cache, sizeof (uuid_node_t));
130                 node_init = 1;
131         }

133         /*
134          * Access the state file, mmap it and initialize the shared lock.
135          * file_type tells us whether we had access to the state file or
136          * created a temporary one.
137          */
138         if (map_state() == -1)
139                 return (-1);

141         /*
142          * Acquire the lock
143          */
144         for (;;) {
145                 if ((ret = mutex_lock(&data->lock)) == 0)
146                         break;
147                 else
148                         switch (ret) {
149                                 case EOWNERDEAD:
150                                         revalidate_data(&system_node);
151                                         (void) mutex_consistent(&data->lock);
152                                         (void) mutex_unlock(&data->lock);
153                                         break;
154                                 case ENOTRECOVERABLE:
155                                         return (ret);
156                         }
157         }

159         /* State file is either new or is temporary, get a random clock seq */
160         if (data->state.clock == 0) {
161                 data->state.clock = get_random();
162                 non_unique++;
163         }

165         if (memcmp(&system_node, &data->state.node, sizeof (uuid_node_t)) != 0)
166                 data->state.clock++;

168         get_current_time(&timestamp);

170         /*
171          * If timestamp is not set or is not in the past, bump
172          * data->state.clock
173          */
174         if ((data->state.ts == 0) || (data->state.ts >= timestamp)) {
175                 data->state.clock++;
176                 data->state.ts = timestamp;
177         }

179         if (non_unique)
180                 system_node.nodeID[0] |= 0x80;

182         /* Stuff fields into the UUID struct */
183         format_uuid(uuid, data->state.clock, timestamp, system_node);

185         (void) mutex_unlock(&data->lock);

187         return (0);
188 }
```



```
190 /*
191  * Fills system_node with Ethernet address if available,
192  * else fills random numbers
193  */
194 static void
195 gen_ethernet_address(uuid_node_t *system_node)
196 {
197         uchar_t         node[6];

199         if (get_ethernet_address(system_node) != 0) {
200                 fill_random_bytes(node, 6);
201                 (void) memcpy(system_node->nodeID, node, 6);
202                 /*
203                  * use 8:0:20 with the multicast bit set
204                  * to avoid namespace collisions.
205                  */
206                 system_node->nodeID[0] = 0x88;
207                 system_node->nodeID[1] = 0x00;
208                 system_node->nodeID[2] = 0x20;
209         }
210 }

212 /*
213  * Formats a UUID, given the clock_seq timestamp, and node address.
214  * Fills in passed-in pointer with the resulting uuid.
215  */
216 static void
217 format_uuid(struct uuid *uuid, uint16_t clock_seq,
218     uuid_time_t timestamp, uuid_node_t node)
219 {

221         /*
222          * First set up the first 60 bits from the timestamp
223          */
224         uuid->time_low = (uint32_t)(timestamp & 0xFFFFFFFF);
225         uuid->time_mid = (uint16_t)((timestamp >> 32) & 0xFFFF);
226         uuid->time_hi_and_version = (uint16_t)((timestamp >> 48) & 0x0FFF);

228         /*
229          * This is version 1, so say so in the UUID version field (4 bits)
230          */
231         uuid->time_hi_and_version |= (1 << 12);

233         /*
234          * Now do the clock sequence
235          */
236         uuid->clock_seq_low = clock_seq & 0xFF;

238         /*
239          * We must save the most-significant 2 bits for the reserved field
240          */
241         uuid->clock_seq_hi_and_reserved = (clock_seq & 0x3F00) >> 8;

243         /*
244          * The variant for this format is the 2 high bits set to 10,
245          * so here it is
246          */
247         uuid->clock_seq_hi_and_reserved |= 0x80;

249         /*
250          * write result to passed-in pointer
251          */
252         (void) memcpy(&uuid->node_addr, &node, sizeof (uuid->node_addr));
253 }

255 /*
```

```
 * Opens/creates the state file, falling back to a tmp
 */
static int
map_state()
{
        FILE    *tmp;

        /* If file's mapped, return */
        if (file_type != 0)
                return (1);

        if ((fd = open(STATE_LOCATION, O_RDWR)) < 0) {
                file_type = TEMP_FILE;

                if ((tmp = tmpfile()) == NULL)
                        return (-1);
                else
                        fd = fileno(tmp);
        } else {
                file_type = STATE_FILE;
        }

        (void) ftruncate(fd, (off_t)sizeof (shared_buffer_t));

        /* LINTED - alignment */
        data = (shared_buffer_t *)mmap(NULL, sizeof (shared_buffer_t),
            PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);

        if (data == MAP_FAILED)
                return (-1);

        (void) mutex_init(&data->lock, USYNC_PROCESS|LOCK_ROBUST, 0);

        (void) close(fd);

        return (1);
}

static void
revalidate_data(uuid_node_t *node)
{
        int i;

        data->state.ts = 0;

        for (i = 0; i < sizeof (data->state.node.nodeID); i++)
                data->state.node.nodeID[i] = 0;

        data->state.clock = 0;

        gen_ethernet_address(node);
        bcopy(node, &node_id_cache, sizeof (uuid_node_t));
        node_init = 1;
}

/*
 * Prints a nicely-formatted uuid to stdout.
 */
void
uuid_print(struct uuid u)
{
        int i;

        (void) printf("%8.8x-%4.4x-%4.4x-%2.2x%2.2x-", u.time_low, u.time_mid,
            u.time_hi_and_version, u.clock_seq_hi_and_reserved,
            u.clock_seq_low);
```



```
        for (i = 0; i < 6; i++)
                (void) printf("%2.2x", u.node_addr[i]);
        (void) printf("\n");
}

/*
 * Only called with urandmtx held.
 * Fills/refills the cache of randomness. We know that our allocations of
 * randomness are always much less than the total size of the cache.
 * Tries to use /dev/urandom random number generator - if that fails for some
 * reason, it retries MAX_RETRY times then sets rcachep to NULL so we no
 * longer use the cache.
 */
static void
load_cache()
{
        int i, retries = 0;
        int nbytes = RCACHE_SIZE;
        char *buf = rcache;

        while (nbytes > 0) {
                i = read(fd_urand, buf, nbytes);
                if ((i < 0) && (errno == EINTR)) {
                        continue;
                }
                if (i <= 0) {
                        if (retries++ == MAX_RETRY)
                                break;
                        continue;
                }
                nbytes -= i;
                buf += i;
                retries = 0;
        }
        if (nbytes == 0)
                rcachep = rcache;
        else
                rcachep = NULL;
}

/*
 * Fills buf with random numbers - nbytes is the number of bytes
 * to fill-in. Tries to use cached data from the /dev/urandom random number
 * generator - if that fails for some reason, it uses srand48(3C)
 */
static void
fill_random_bytes(uchar_t *buf, int nbytes)
{
        int i;

        if (fd_urand == -1) {
                (void) mutex_lock(&urandmtx);
                /* check again now that we have the mutex */
                if (fd_urand == -1) {
                        if ((fd_urand = open(URANDOM_PATH, O_RDONLY)) >= 0)
                                load_cache();
                }
                (void) mutex_unlock(&urandmtx);
        }
        if (fd_urand >= 0 && rcachep != NULL) {
                int cnt;

                (void) mutex_lock(&urandmtx);
                if (rcachep != NULL &&
                    (rcachep + nbytes) >= (rcache + RCACHE_SIZE))
                        load_cache();
```

```
389                 if (rcachep != NULL) {
390                         for (cnt = 0; cnt < nbytes; cnt++)
391                                 *buf++ = *rcachep++;
392                         (void) mutex_unlock(&urandmtx);
393                         return;
394                 }
395                 (void) mutex_unlock(&urandmtx);
396         }
397         for (i = 0; i < nbytes; i++) {
398                 *buf++ = get_random() & 0xFF;
399         }
400 }

402 /*
403  * Unpacks the structure members in "struct uuid" to a char string "uuid_t".
404  */
405 void
406 struct_to_string(uuid_t ptr, struct uuid *uu)
407 {
408         uint_t          tmp;
409         uchar_t         *out = ptr;

411         tmp = uu->time_low;
412         out[3] = (uchar_t)tmp;
413         tmp >>= 8;
414         out[2] = (uchar_t)tmp;
415         tmp >>= 8;
416         out[1] = (uchar_t)tmp;
417         tmp >>= 8;
418         out[0] = (uchar_t)tmp;

420         tmp = uu->time_mid;
421         out[5] = (uchar_t)tmp;
422         tmp >>= 8;
423         out[4] = (uchar_t)tmp;

425         tmp = uu->time_hi_and_version;
426         out[7] = (uchar_t)tmp;
427         tmp >>= 8;
428         out[6] = (uchar_t)tmp;

430         tmp = uu->clock_seq_hi_and_reserved;
431         out[8] = (uchar_t)tmp;
432         tmp = uu->clock_seq_low;
433         out[9] = (uchar_t)tmp;

435         (void) memcpy(out+10, uu->node_addr, 6);

437 }

439 /*
440  * Packs the values in the "uuid_t" string into "struct uuid".
441  */
442 void
443 string_to_struct(struct uuid *uuid, uuid_t in)
444 {
445         uchar_t *ptr;
446         uint_t  tmp;

448         ptr = in;

450         tmp = *ptr++;
451         tmp = (tmp << 8) | *ptr++;
452         tmp = (tmp << 8) | *ptr++;
453         tmp = (tmp << 8) | *ptr++;
```



```
454         uuid->time_low = tmp;

456         tmp = *ptr++;
457         tmp = (tmp << 8) | *ptr++;
458         uuid->time_mid = tmp;

460         tmp = *ptr++;
461         tmp = (tmp << 8) | *ptr++;
462         uuid->time_hi_and_version = tmp;

464         tmp = *ptr++;
465         uuid->clock_seq_hi_and_reserved = tmp;

467         tmp = *ptr++;
468         uuid->clock_seq_low = tmp;

470         (void) memcpy(uuid->node_addr, ptr, 6);

472 }

474 /*
475  * Generates UUID based on DCE Version 4
476  */
477 void
478 uuid_generate_random(uuid_t uu)
479 {
480         struct uuid     uuid;

482         if (uu == NULL)
483                 return;

485         (void) memset(uu, 0, sizeof (uuid_t));
486         (void) memset(&uuid, 0, sizeof (struct uuid));

488         fill_random_bytes(uu, sizeof (uuid_t));
489         string_to_struct(&uuid, uu);
490         /*
491          * This is version 4, so say so in the UUID version field (4 bits)
492          */
493         uuid.time_hi_and_version |= (1 << 14);
494         /*
495          * we don’t want the bit 1 to be set also which is for version 1
496          */
497         uuid.time_hi_and_version &= VER1_MASK;

499         /*
500          * The variant for this format is the 2 high bits set to 10,
501          * so here it is
502          */
503         uuid.clock_seq_hi_and_reserved |= 0x80;

505         /*
506          * Set MSB of Ethernet address to 1 to indicate that it was generated
507          * randomly
508          */
509         uuid.node_addr[0] |= 0x80;
510         struct_to_string(uu, &uuid);
511 }

513 /*
514  * Generates UUID based on DCE Version 1.
515  */
516 void
517 uuid_generate_time(uuid_t uu)
518 {
519         struct  uuid uuid;
```

```
521         if (uu == NULL)
522                 return;

524         if (uuid_create(&uuid) < 0) {
525                 uuid_generate_random(uu);
526                 return;
527         }

529         struct_to_string(uu, &uuid);
530 }

532 /*
533  * Creates a new UUID. The uuid will be generated based on high-quality
534  * randomness from /dev/urandom, if available by calling uuid_generate_random.
535  * If it failed to generate UUID then uuid_generate will call
536  * uuid_generate_time.
537  */
538 void
539 uuid_generate(uuid_t uu)
540 {
541         if (uu == NULL) {
542                 return;
543         }
544         if (fd_urand == -1) {
545                 (void) mutex_lock(&urandmtx);
546                 /* check again now that we have the mutex */
547                 if (fd_urand == -1) {
548                         if ((fd_urand = open(URANDOM_PATH, O_RDONLY)) >= 0)
549                                 load_cache();
550                 }
551                 (void) mutex_unlock(&urandmtx);
552         }
553         if (fd_urand >= 0) {
554                 uuid_generate_random(uu);
555         } else {
556                 (void) uuid_generate_time(uu);
557         }
558 }

560 /*
561  * Copies the UUID variable src to dst.
562  */
563 void
564 uuid_copy(uuid_t dst, uuid_t src)
565 {
566         (void) memcpy(dst, src, UUID_LEN);
567 }

569 /*
570  * Sets the value of the supplied uuid variable uu, to the NULL value.
571  */
572 void
573 uuid_clear(uuid_t uu)
574 {
575         (void) memset(uu, 0, UUID_LEN);
576 }

578 /*
579  * This function converts the supplied UUID uu from the internal
580  * binary format into a 36-byte string (plus trailing null char)
581  * and stores this value in the character string pointed to by out.
582  */
583 static void
584 uuid_unparse_common(uuid_t uu, char *out, boolean_t upper)
 26 void
```



```
 27 uuid_unparse(uuid_t uu, char *out)
585 {
586         struct uuid     uuid;
587         uint16_t        clock_seq;
588         char            etheraddr[13];
589         int             index = 0, i;

591         /* basic sanity checking */
592         if (uu == NULL) {
593                 return;
594         }

 39         /* XXX user should have allocated enough memory */
 40         /*
 41          * if (strlen(out) < UUID_PRINTABLE_STRING_LENGTH) {
 42          * return;
 43          * }
 44          */
596         string_to_struct(&uuid, uu);
597         clock_seq = uuid.clock_seq_hi_and_reserved;
598         clock_seq = (clock_seq  << 8) | uuid.clock_seq_low;
599         for (i = 0; i < 6; i++) {
600                 (void) sprintf(&etheraddr[index++], upper ? "%.2X" : "%.2x",
601                     uuid.node_addr[i]);
 49                 (void) sprintf(&etheraddr[index++], "%.2x", uuid.node_addr[i]);
602                 index++;
603         }
604         etheraddr[index] = '\0';

606         (void) snprintf(out, 25,
607             upper ? "%08X-%04X-%04X-%04X-" : "%08x-%04x-%04x-%04x-",
 54         (void) snprintf(out, 25, "%08x-%04x-%04x-%04x-",
608             uuid.time_low, uuid.time_mid, uuid.time_hi_and_version, clock_seq);
609         (void) strlcat(out, etheraddr, UUID_PRINTABLE_STRING_LENGTH);
610 }

612 void
613 uuid_unparse_upper(uuid_t uu, char *out)
614 {
615         uuid_unparse_common(uu, out, B_TRUE);
616 }

618 void
619 uuid_unparse_lower(uuid_t uu, char *out)
620 {
621         uuid_unparse_common(uu, out, B_FALSE);
622 }

624 void
625 uuid_unparse(uuid_t uu, char *out)
626 {
627         /*
628          * Historically uuid_unparse on Solaris returns lower case,
629          * for compatibility we preserve this behaviour.
630          */
631         uuid_unparse_common(uu, out, B_FALSE);
632 }

634 #endif /* ! codereview */
635 /*
636  * The uuid_is_null function compares the value of the supplied
637  * UUID variable uu to the NULL value. If the value is equal
638  * to the NULL UUID, 1 is returned, otherwise 0 is returned.
639  */
640 int
641 uuid_is_null(uuid_t uu)
```

```
642 {
643         int             i;
644         uuid_t          null_uu;

646         (void) memset(null_uu, 0, sizeof (uuid_t));
647         i = memcmp(uu, null_uu, sizeof (uuid_t));
648         if (i == 0) {
649                 /* uu is NULL uuid */
650                 return (1);
651         } else {
652                 return (0);
653         }
654 }

656 /*
657  * uuid_parse converts the UUID string given by 'in' into the
658  * internal uuid_t format. The input UUID is a string of the form
659  * cefa7a9c-1dd2-11b2-8350-880020adbeef in printf(3C) format.
660  * Upon successfully parsing the input string, UUID is stored
661  * in the location pointed to by uu
662  */
663 int
664 uuid_parse(char *in, uuid_t uu)
665 {

667         char            *ptr, buf[3];
668         int             i;
669         struct uuid     uuid;
670         uint16_t        clock_seq;

672         /* do some sanity checking */
673         if ((strlen(in) != 36) || (uu == NULL) || (in[36] != '\0')) {
674                 return (-1);
675         }

677         ptr = in;
678         for (i = 0; i < 36; i++, ptr++) {
679                 if ((i == 8) || (i == 13) || (i == 18) || (i == 23)) {
680                         if (*ptr != '-') {
681                                 return (-1);
682                         }
683                 } else {
684                         if (!isxdigit(*ptr)) {
685                                 return (-1);
686                         }
687                 }
688         }

690         uuid.time_low = strtoul(in, NULL, 16);
691         uuid.time_mid = strtoul(in+9, NULL, 16);
692         uuid.time_hi_and_version = strtoul(in+14, NULL, 16);
693         clock_seq = strtoul(in+19, NULL, 16);
694         uuid.clock_seq_hi_and_reserved = (clock_seq & 0xFF00) >> 8;
695         uuid.clock_seq_low = (clock_seq & 0xFF);

697         ptr = in+24;
698         buf[2] = '\0';
699         for (i = 0; i < 6; i++) {
700                 buf[0] = *ptr++;
701                 buf[1] = *ptr++;
702                 uuid.node_addr[i] = strtoul(buf, NULL, 16);
703         }
704         struct_to_string(uu, &uuid);
705         return (0);
706 }
```



```
708 /*
709  * uuid_time extracts the time at which the supplied UUID uu
710  * was created. This function can only extract the creation
711  * time for UUIDs created with the uuid_generate_time function.
712  * The time at which the UUID was created, in seconds and
713  * microseconds since the epoch is stored in the location
714  * pointed to by ret_tv.
715  */
716 time_t
717 uuid_time(uuid_t uu, struct timeval *ret_tv)
718 {
719         struct uuid     uuid;
720         uint_t          high;
721         struct timeval  tv;
722         u_longlong_t    clock_reg;
723         uint_t          tmp;
724         uint8_t         clk;

726         string_to_struct(&uuid, uu);
727         tmp = (uuid.time_hi_and_version & 0xF000) >> 12;
728         clk = uuid.clock_seq_hi_and_reserved;

730         /* check if uu is NULL, Version = 1 of DCE and Variant = 0b10x */
731         if ((uu == NULL) || ((tmp & 0x01) != 0x01) || ((clk & 0x80) != 0x80)) {
732                 return (-1);
733         }
734         high = uuid.time_mid | ((uuid.time_hi_and_version & 0xFFF) << 16);
735         clock_reg = uuid.time_low | ((u_longlong_t)high << 32);

737         clock_reg -= (((u_longlong_t)0x01B21DD2) << 32) + 0x13814000;
738         tv.tv_sec = clock_reg / 10000000;
739         tv.tv_usec = (clock_reg % 10000000) / 10;

741         if (ret_tv) {
742                 *ret_tv = tv;
743         }

745         return (tv.tv_sec);
746 }
```

****
1557 Wed Apr 9 02:10:57 2014
new/usr/src/man/man3uuid/Makefile
4118 libuuid should provide uuid_unparse_{upper,lower} functions
Reviewed by: Serghei Samsi <sscdvp@gmail.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
****

```
#
# 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 2014 Andrew Stormont.
#endif /* ! codereview */
#

include         $(SRC)/Makefile.master

MANSECT=        3uuid

MANFILES=       uuid_clear.3uuid

MANLINKS=       uuid_compare.3uuid              \
                uuid_copy.3uuid                 \
                uuid_generate.3uuid             \
                uuid_generate_random.3uuid      \
                uuid_generate_time.3uuid        \
                uuid_is_null.3uuid              \
                uuid_parse.3uuid                \
                uuid_time.3uuid                 \
                uuid_unparse.3uuid              \
                uuid_unparse_lower.3uuid        \
                uuid_unparse_upper.3uuid
                uuid_unparse.3uuid

uuid_compare.3uuid              := LINKSRC = uuid_clear.3uuid
uuid_copy.3uuid                 := LINKSRC = uuid_clear.3uuid
uuid_generate.3uuid             := LINKSRC = uuid_clear.3uuid
uuid_generate_random.3uuid      := LINKSRC = uuid_clear.3uuid
uuid_generate_time.3uuid        := LINKSRC = uuid_clear.3uuid
uuid_is_null.3uuid              := LINKSRC = uuid_clear.3uuid
uuid_parse.3uuid                := LINKSRC = uuid_clear.3uuid
uuid_time.3uuid                 := LINKSRC = uuid_clear.3uuid
uuid_unparse.3uuid              := LINKSRC = uuid_clear.3uuid
uuid_unparse_lower.3uuid        := LINKSRC = uuid_clear.3uuid
uuid_unparse_upper.3uuid        := LINKSRC = uuid_clear.3uuid
#endif /* ! codereview */

.KEEP_STATE:

include         $(SRC)/man/Makefile.man

install:        $(ROOTMANFILES) $(ROOTMANLINKS)
```

```
**********************************************************
    6061 Wed Apr  9 02:10:57 2014
new/usr/src/man/man3uuid/uuid_clear.3uuid
4118 libuuid should provide uuid_unparse_{upper,lower} functions
Reviewed by: Serghei Samsi <sscdvp@gmail.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
'\" te
.\" Copyright (c) 2006, Sun Microsystems, Inc. All Rights Reserved.
.\" Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH UUID_CLEAR 3UUID "Apr 9, 2014"
.TH UUID_CLEAR 3UUID "Jan 16, 2006"
.SH NAME
uuid_clear, uuid_compare, uuid_copy, uuid_generate, uuid_generate_random,
uuid_generate_time, uuid_is_null, uuid_parse, uuid_time, uuid_unparse,
uuid_unparse_lower, uuid_unparse_upper \- universally unique identifier (UUID)
operations
uuid_generate_time, uuid_is_null, uuid_parse, uuid_time, uuid_unparse \-
universally unique identifier (UUID) operations
.SH SYNOPSIS
.LP
.nf
cc [ \fIflag \&.\|.\|.\fR ] \fIfile\fR\&.\|.\|. \fB-luuid\fR [ \fIlibrary \&.\|.
#include <uuid/uuid.h>

\fBvoid\fR \fBuuid_clear\fR(\fBuuid_t\fR \fIuu\fR);
.fi

.LP
.nf
\fBint\fR \fBuuid_compare\fR(\fBuuid_t\fR \fIuu1\fR, \fBuuid_t\fR \fIuu2\fR);
.fi

.LP
.nf
\fBvoid\fR \fBuuid_copy\fR(\fBuuid_t\fR \fIdst\fR, \fBuuid_t\fR \fIsrc\fR);
.fi

.LP
.nf
\fBvoid\fR \fBuuid_generate\fR(\fBuuid_t\fR \fIout\fR);
.fi

.LP
.nf
\fBvoid\fR \fBuuid_generate_random\fR(\fBuuid_t\fR \fIout\fR);
.fi

.LP
.nf
\fBvoid\fR \fBuuid_generate_time\fR(\fBuuid_t\fR \fIout\fR);
.fi

.LP
.nf
\fBint\fR \fBuuid_is_null\fR(\fBuuid_t\fR \fIuu\fR);
.fi

.LP
.nf
```



```
\fBint\fR \fBuuid_parse\fR(\fBchar *\fR\fIin\fR, \fBuuid_t\fR \fIuu\fR);
.fi

.LP
.nf
\fBtime_t\fR \fBuuid_time\fR(\fBuuid_t\fR \fIuu\fR, \fBstruct timeval *\fR\fIret
.fi

.LP
.nf
\fBvoid\fR \fBuuid_unparse\fR(\fBuuid_t\fR \fIuu\fR, \fBchar *\fR\fIout\fR);
.fi

.LP
.nf
\fBvoid\fR \fBuuid_unparse_lower\fR(\fBuuid_t\fR \fIuu\fR, \fBchar *\fR\fIout\fR
.fi

.LP
.nf
\fBvoid\fR \fBuuid_unparse_upper\fR(\fBuuid_t\fR \fIuu\fR, \fBchar *\fR\fIout\fR
.fi

#endif /* ! codereview */
.SH DESCRIPTION
.sp
.LP
The \fBuuid_clear()\fR function sets the value of the specified universally
unique identifier (UUID) variable \fIuu\fR to the \fINULL\fR value.
.sp
.LP
The \fBuuid_compare()\fR function compares the two specified UUID variables
\fIuu1\fR and \fIuu2\fR to each other. It returns an integer less than, equal
to, or greater than zero if \fIuu1\fR is found to be, respectively,
lexicographically less than, equal, or greater than \fIuu2\fR.
.sp
.LP
The \fBuuid_copy()\fR function copies the UUID variable \fIsrc\fR to \fIdst\fR.
.sp
.LP
The \fBuuid_generate()\fR function creates a new UUID that is generated based
on high-quality randomness from \fB/dev/urandom\fR, if available. If
\fB/dev/urandom\fR is not available, \fBuuid_generate()\fR calls
\fBuuid_generate_time()\fR. Because the use of this algorithm provides
information about when and where the UUID was generated, it could cause privacy
problems for some applications.
.sp
.LP
The \fBuuid_generate_random()\fR function produces a UUID with a random or
pseudo-randomly generated time and Ethernet MAC address that corresponds to a
DCE version 4 UUID.
.sp
.LP
The \fBuuid_generate_time()\fR function uses the current time and the local
Ethernet MAC address (if available, otherwise a MAC address is fabricated) that
corresponds to a DCE version 1 UUID. If the UUID is not guaranteed to be
unique, the multicast bit is set (the high-order bit of octet number 10).
.sp
.LP
The \fBuuid_is_null()\fR function compares the value of the specified UUID
variable \fIuu\fR to the \fINULL\fR value. If the value is equal to the
\fINULL\fR UUID, 1 is returned. Otherwise 0 is returned.
.sp
.LP
The \fBuuid_parse()\fR function converts the UUID string specified by \fIin\fR
to the internal \fBuuid_t\fR format. The input UUID is a string of the form
```

\fBcefa7a9c-1dd2-11b2-8350-880020adbeef\fR. In \fBprintf\fR(3C) format, the string is "\fB%08x-%04x-%04x-%04x-%012x\fR", 36 bytes plus the trailing null character. If the input string is parsed successfully, \fB0\fR is returned and the UUID is stored in the location pointed to by \fIuu\fR. Otherwise \fB-1\fR is returned.

.sp

.LP

The \fBuuid_time()\fR function extracts the time at which the specified UUID \fIuu\fR was created. Since the UUID creation time is encoded within the UUID, this function can reasonably be expected to extract the creation time only for UUIDs created with the \fBuuid_generate_time()\fR function. The time at which the UUID was created, in seconds since January 1, 1970 GMT (the epoch), is returned (see \fBtime\fR(2)). The time at which the UUID was created, in seconds and microseconds since the epoch is also stored in the location pointed to by \fBret_tv\fR (see \fBgettimeofday\fR(3C)).

.sp

.LP

The \fBuuid_unparse()\fR and \fBuuid_unparse_lower()\fR functions convert the specified UUID \fIuu\fR from the internal binary format to a lower case string of the length defined in the \fBuuid.h\fR macro, \fBUUID_PRINTABLE_STRING_LENGTH\fR, which includes the trailing null character. The resulting value is stored in the character string pointed to by \fIout\fR.

.sp

.LP

The \fBuuid_unparse_upper()\fR function converts the specified UUID \fIuu\fR from the internal binary format to a upper case string of the length defined in the \fBuuid.h\fR macro, \fBUUID_PRINTABLE_STRING_LENGTH\fR, which includes the trailing null character. The resulting value is stored in the character string pointed to by \fIout\fR.

The \fBuuid_unparse()\fR function converts the specified UUID \fIuu\fR from the internal binary format to a string of the length defined in the \fBuuid.h\fR macro, \fBUUID_PRINTABLE_STRING_LENGTH\fR, which includes the trailing null character. The resulting value is stored in the character string pointed to by \fIout\fR.

.SH ATTRIBUTES

.sp

.LP

See \fBattributes\fR(5) for descriptions of the following attributes:

.sp

.sp

.TS

box;

c | c

l | l .

ATTRIBUTE TYPE ATTRIBUTE VALUE

_

Interface Stability Evolving

_

MT-Level Safe

.TE

.SH SEE ALSO

.sp

.LP

\fBinetd\fR(1M), \fBtime\fR(2), \fBgettimeofday\fR(3C), \fBlibuuid\fR(3LIB), \fBprintf\fR(3C), \fBattributes\fR(5)

****
1462 Wed Apr 9 02:10:58 2014
new/usr/src/pkg/manifests/system-library.man3uuid.inc
4118 libuuid should provide uuid_unparse_{upper,lower} functions
Reviewed by: Serghei Samsi <sscdvp@gmail.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
****

```
   1 #
   2 # This file and its contents are supplied under the terms of the
   3 # Common Development and Distribution License ("CDDL"), version 1.0.
   4 # You may only use this file in accordance with the terms of version
   5 # 1.0 of the CDDL.
   6 #
   7 # A full copy of the text of the CDDL should have accompanied this
   8 # source.  A copy of the CDDL is also available via the Internet
   9 # at http://www.illumos.org/license/CDDL.
  10 #

  12 #
  13 # Copyright 2011, Richard Lowe
  14 # Copyright 2012 Nexenta Systems, Inc. All rights reserved.
  15 # Copyright 2014 Andrew Stormont.
  16 #endif /* ! codereview */
  17 #

  19 file path=usr/share/man/man3uuid/uuid_clear.3uuid
  20 link path=usr/share/man/man3uuid/uuid_compare.3uuid target=uuid_clear.3uuid
  21 link path=usr/share/man/man3uuid/uuid_copy.3uuid target=uuid_clear.3uuid
  22 link path=usr/share/man/man3uuid/uuid_generate.3uuid target=uuid_clear.3uuid
  23 link path=usr/share/man/man3uuid/uuid_generate_random.3uuid \
  24     target=uuid_clear.3uuid
  25 link path=usr/share/man/man3uuid/uuid_generate_time.3uuid \
  26     target=uuid_clear.3uuid
  27 link path=usr/share/man/man3uuid/uuid_is_null.3uuid target=uuid_clear.3uuid
  28 link path=usr/share/man/man3uuid/uuid_parse.3uuid target=uuid_clear.3uuid
  29 link path=usr/share/man/man3uuid/uuid_time.3uuid target=uuid_clear.3uuid
  30 link path=usr/share/man/man3uuid/uuid_unparse.3uuid target=uuid_clear.3uuid
  31 link path=usr/share/man/man3uuid/uuid_unparse_lower.3uuid \
  32     target=uuid_clear.3uuid
  33 link path=usr/share/man/man3uuid/uuid_unparse_upper.3uuid \
  34     target=uuid_clear.3uuid
  35 #endif /* ! codereview */
```