```
**********************************************************
    8725 Mon Sep 10 00:05:32 2012
new/usr/src/cmd/localedef/ctype.c
3154 Nonconforming tolower and toupper with UTF-8 locales
Reviewed by: Garrett D'Amore <garrett.damore@gmail.com>
**********************************************************
______unchanged_portion_omitted_

 205 void
 206 dump_ctype(void)
 207 {
 208         FILE            *f;
 209         _FileRuneLocale rl;
 210         ctype_node_t    *ctn, *last_ct, *last_lo, *last_up;
 211         _FileRuneEntry  *ct = NULL;
 212         _FileRuneEntry  *lo = NULL;
 213         _FileRuneEntry  *up = NULL;
 214         wchar_t         wc;
 215 #endif /* ! codereview */

 217         (void) memset(&rl, 0, sizeof (rl));
 218         last_ct = NULL;
 219         last_lo = NULL;
 220         last_up = NULL;

 222         if ((f = open_category()) == NULL)
 223                 return;

 225         (void) memcpy(rl.magic, _FILE_RUNE_MAGIC_1, 8);
 226         (void) strncpy(rl.encoding, get_wide_encoding(), sizeof (rl.encoding));

 228         /*
 229          * Initialize the identity map.
 230          */
 231         for (wc = 0; (unsigned)wc < _CACHED_RUNES; wc++) {
 232                 rl.maplower[wc] = wc;
 233                 rl.mapupper[wc] = wc;
 234         }

 236 #endif /* ! codereview */
 237         for (ctn = avl_first(&ctypes); ctn; ctn = AVL_NEXT(&ctypes, ctn)) {
 238                 int conflict = 0;
 239 #endif /* ! codereview */

 241                 wc = ctn->wc;
 214                 wchar_t wc = ctn->wc;
 215                 int conflict = 0;

 243                 /*
 244                  * POSIX requires certain portable characters have
 245                  * certain types.  Add them if they are missing.
 246                  */
 247                 if ((wc >= 1) && (wc <= 127)) {
 248                         if ((wc >= 'A') && (wc <= 'Z'))
 249                                 ctn->ctype |= _ISUPPER;
 250                         if ((wc >= 'a') && (wc <= 'z'))
 251                                 ctn->ctype |= _ISLOWER;
 252                         if ((wc >= '0') && (wc <= '9'))
 253                                 ctn->ctype |= _ISDIGIT;
 254                         if (strchr(" \f\n\r\t\v", (char)wc) != NULL)
 255                                 ctn->ctype |= _ISSPACE;
 256                         if (strchr("0123456789ABCDEFabcdef", (char)wc) != NULL)
 257                                 ctn->ctype |= _ISXDIGIT;
 258                         if (strchr(" \t", (char)wc))
 259                                 ctn->ctype |= _ISBLANK;
```



```
 261                         /*
 262                          * Technically these settings are only
 263                          * required for the C locale.  However, it
 264                          * turns out that because of the historical
 265                          * version of isprint(), we need them for all
 266                          * locales as well.  Note that these are not
 267                          * necessarily valid punctation characters in
 268                          * the current language, but ispunct() needs
 269                          * to return TRUE for them.
 270                          */
 271                         if (strchr("!\"'#$%&()*+,-./:;<=>?@[\\]^_'{|}~",
 272                             (char)wc))
 273                                 ctn->ctype |= _ISPUNCT;
 274                 }

 276                 /*
 277                  * POSIX also requires that certain types imply
 278                  * others.  Add any inferred types here.
 279                  */
 280                 if (ctn->ctype & (_ISUPPER |_ISLOWER))
 281                         ctn->ctype |= _ISALPHA;
 282                 if (ctn->ctype & _ISDIGIT)
 283                         ctn->ctype |= _ISXDIGIT;
 284                 if (ctn->ctype & _ISBLANK)
 285                         ctn->ctype |= _ISSPACE;
 286                 if (ctn->ctype & (_ISALPHA|_ISDIGIT|_ISXDIGIT))
 287                         ctn->ctype |= _ISGRAPH;
 288                 if (ctn->ctype & _ISGRAPH)
 289                         ctn->ctype |= _ISPRINT;

 291                 /*
 292                  * Finally, POSIX requires that certain combinations
 293                  * are invalid.  We don't flag this as a fatal error,
 294                  * but we will warn about.
 295                  */
 296                 if ((ctn->ctype & _ISALPHA) &&
 297                     (ctn->ctype & (_ISPUNCT|_ISDIGIT)))
 298                         conflict++;
 299                 if ((ctn->ctype & _ISPUNCT) &
 300                     (ctn->ctype & (_ISDIGIT|_ISALPHA|_ISXDIGIT)))
 301                         conflict++;
 302                 if ((ctn->ctype & _ISSPACE) && (ctn->ctype & _ISGRAPH))
 303                         conflict++;
 304                 if ((ctn->ctype & _ISCNTRL) & _ISPRINT)
 305                         conflict++;
 306                 if ((wc == ' ') && (ctn->ctype & (_ISPUNCT|_ISGRAPH)))
 307                         conflict++;

 309                 if (conflict) {
 310                         warn("conflicting classes for character 0x%x (%x)",
 311                             wc, ctn->ctype);
 312                 }
 313                 /*
 314                  * Handle the lower 256 characters using the simple
 315                  * optimization.  Note that if we have not defined the
 316                  * upper/lower case, then we identity map it.
 317                  */
 318                 if ((unsigned)wc < _CACHED_RUNES) {
 319                         rl.runetype[wc] = ctn->ctype;
 320                         if (ctn->tolower)
 321                                 rl.maplower[wc] = ctn->tolower;
 322                         if (ctn->toupper)
 323                                 rl.mapupper[wc] = ctn->toupper;
 294                         rl.maplower[wc] = ctn->tolower ? ctn->tolower : wc;
 295                         rl.mapupper[wc] = ctn->toupper ? ctn->toupper : wc;
 324                         continue;
```

```
325                 }

327                 if ((last_ct != NULL) && (last_ct->ctype == ctn->ctype)) {
328                         ct[rl.runetype_ext_nranges-1].max = wc;
329                         last_ct = ctn;
330                 } else {
331                         rl.runetype_ext_nranges++;
332                         ct = realloc(ct,
333                             sizeof (*ct) * rl.runetype_ext_nranges);
334                         ct[rl.runetype_ext_nranges - 1].min = wc;
335                         ct[rl.runetype_ext_nranges - 1].max = wc;
336                         ct[rl.runetype_ext_nranges - 1].map = ctn->ctype;
337                         last_ct = ctn;
338                 }
339                 if (ctn->tolower == 0) {
340                         last_lo = NULL;
341                 } else if ((last_lo != NULL) &&
342                     (last_lo->tolower + 1 == ctn->tolower)) {
343                         lo[rl.maplower_ext_nranges-1].max = wc;
344                         last_lo = ctn;
345                 } else {
346                         rl.maplower_ext_nranges++;
347                         lo = realloc(lo,
348                             sizeof (*lo) * rl.maplower_ext_nranges);
349                         lo[rl.maplower_ext_nranges - 1].min = wc;
350                         lo[rl.maplower_ext_nranges - 1].max = wc;
351                         lo[rl.maplower_ext_nranges - 1].map = ctn->tolower;
352                         last_lo = ctn;
353                 }

355                 if (ctn->toupper == 0) {
356                         last_up = NULL;
357                 } else if ((last_up != NULL) &&
358                     (last_up->toupper + 1 == ctn->toupper)) {
359                         up[rl.mapupper_ext_nranges-1].max = wc;
360                         last_up = ctn;
361                 } else {
362                         rl.mapupper_ext_nranges++;
363                         up = realloc(up,
364                             sizeof (*up) * rl.mapupper_ext_nranges);
365                         up[rl.mapupper_ext_nranges - 1].min = wc;
366                         up[rl.mapupper_ext_nranges - 1].max = wc;
367                         up[rl.mapupper_ext_nranges - 1].map = ctn->toupper;
368                         last_up = ctn;
369                 }
370         }

372         if ((wr_category(&rl, sizeof (rl), f) < 0) ||
373             (wr_category(ct, sizeof (*ct) * rl.runetype_ext_nranges, f) < 0) ||
374             (wr_category(lo, sizeof (*lo) * rl.maplower_ext_nranges, f) < 0) ||
375             (wr_category(up, sizeof (*up) * rl.mapupper_ext_nranges, f) < 0)) {
376                 return;
377         }

379         close_category(f);
380 }
```

***unchanged_portion_omitted_***