```
**********************************************************
   26278 Sun Mar 29 19:44:23 2015
new/usr/src/head/unistd.h
5776 vfork and getwd should not be exposed under XPG7
**********************************************************
```

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

  22 /*
  23  * Copyright 2014 Garrett D'Amore <garrett@damore.org>
  24  * Copyright (c) 2013 Gary Mills
  25  *
  26  * Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
  27  */

  29 /*      Copyright (c) 1988 AT&T */
  30 /*        All Rights Reserved   */

  32 /* Copyright (c) 2013, OmniTI Computer Consulting, Inc. All rights reserved. */

  34 #ifndef _UNISTD_H
  35 #define _UNISTD_H

  37 #include <sys/feature_tests.h>

  39 #include <sys/types.h>
  40 #include <sys/unistd.h>

  42 #ifdef  __cplusplus
  43 extern "C" {
  44 #endif

  46 /* Symbolic constants for the "access" routine: */
  47 #define R_OK    4       /* Test for Read permission */
  48 #define W_OK    2       /* Test for Write permission */
  49 #define X_OK    1       /* Test for eXecute permission */
  50 #define F_OK    0       /* Test for existence of File */

  52 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || defined(__EXTENSIONS__)
  53 #define F_ULOCK 0       /* Unlock a previously locked region */
  54 #define F_LOCK  1       /* Lock a region for exclusive use */
  55 #define F_TLOCK 2       /* Test and lock a region for exclusive use */
  56 #define F_TEST  3       /* Test a region for other processes locks */
  57 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2)... */

  59 /* Symbolic constants for the "lseek" routine: */

  61 #ifndef SEEK_SET
```



```
  62 #define SEEK_SET        0       /* Set file pointer to "offset" */
  63 #endif

  65 #ifndef SEEK_CUR
  66 #define SEEK_CUR        1       /* Set file pointer to current plus "offset" */
  67 #endif

  69 #ifndef SEEK_END
  70 #define SEEK_END        2       /* Set file pointer to EOF plus "offset" */
  71 #endif

  73 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
  74 #ifndef SEEK_DATA
  75 #define SEEK_DATA       3       /* Set file pointer to next data past offset */
  76 #endif

  78 #ifndef SEEK_HOLE
  79 #define SEEK_HOLE       4       /* Set file pointer to next hole past offset */
  80 #endif
  81 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */

  83 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
  84 /* Path names: */
  85 #define GF_PATH "/etc/group"    /* Path name of the "group" file */
  86 #define PF_PATH "/etc/passwd"   /* Path name of the "passwd" file */
  87 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */

  89 /*
  90  * compile-time symbolic constants,
  91  * Support does not mean the feature is enabled.
  92  * Use pathconf/sysconf to obtain actual configuration value.
  93  */

  95 /* Values unchanged in UNIX 03 */
  96 #define _POSIX_ASYNC_IO                 1
  97 #define _POSIX_JOB_CONTROL              1
  98 #define _POSIX_SAVED_IDS                1
  99 #define _POSIX_SYNC_IO                  1

 101 /*
 102  * POSIX.1b compile-time symbolic constants.
 103  */
 104 #if defined(_XPG6)
 105 #define _POSIX_ASYNCHRONOUS_IO          200112L
 106 #define _POSIX_FSYNC                    200112L
 107 #define _POSIX_MAPPED_FILES             200112L
 108 #define _POSIX_MEMLOCK                  200112L
 109 #define _POSIX_MEMLOCK_RANGE            200112L
 110 #define _POSIX_MEMORY_PROTECTION        200112L
 111 #define _POSIX_MESSAGE_PASSING          200112L
 112 #define _POSIX_PRIORITY_SCHEDULING      200112L
 113 #define _POSIX_REALTIME_SIGNALS         200112L
 114 #define _POSIX_SEMAPHORES               200112L
 115 #define _POSIX_SHARED_MEMORY_OBJECTS    200112L
 116 #define _POSIX_SYNCHRONIZED_IO          200112L
 117 #else
 118 #define _POSIX_ASYNCHRONOUS_IO          1
 119 #define _POSIX_FSYNC                    1
 120 #define _POSIX_MAPPED_FILES             1
 121 #define _POSIX_MEMLOCK                  1
 122 #define _POSIX_MEMLOCK_RANGE            1
 123 #define _POSIX_MEMORY_PROTECTION        1
 124 #define _POSIX_MESSAGE_PASSING          1
 125 #define _POSIX_PRIORITY_SCHEDULING      1
 126 #define _POSIX_REALTIME_SIGNALS         1
 127 #define _POSIX_SEMAPHORES               1
```

```
128 #define _POSIX_SHARED_MEMORY_OBJECTS    1
129 #define _POSIX_SYNCHRONIZED_IO          1
130 #endif

132 /*
133  * POSIX.1c compile-time symbolic constants.
134  */
135 #if defined(_XPG6)
136 #define _POSIX_THREAD_SAFE_FUNCTIONS            200112L
137 #define _POSIX_THREADS                          200112L
138 #define _POSIX_THREAD_ATTR_STACKADDR            200112L
139 #define _POSIX_THREAD_ATTR_STACKSIZE            200112L
140 #define _POSIX_THREAD_PROCESS_SHARED            200112L
141 #define _POSIX_THREAD_PRIORITY_SCHEDULING       200112L
142 #define _POSIX_TIMERS                           200112L
143 #else
144 #define _POSIX_THREAD_SAFE_FUNCTIONS            1
145 #define _POSIX_THREADS                          1
146 #define _POSIX_THREAD_ATTR_STACKADDR            1
147 #define _POSIX_THREAD_ATTR_STACKSIZE            1
148 #define _POSIX_THREAD_PROCESS_SHARED            1
149 #define _POSIX_THREAD_PRIORITY_SCHEDULING       1
150 #define _POSIX_TIMERS                           1
151 #endif

153 /* New in UNIX 03 */
154 #define _POSIX_ADVISORY_INFO                    200112L
155 #define _POSIX_BARRIERS                         200112L
156 #define _POSIX_CLOCK_SELECTION                  200112L
157 #define _POSIX_IPV6                             200112L
158 #define _POSIX_MONOTONIC_CLOCK                  200112L
159 #define _POSIX_RAW_SOCKETS                      200112L
160 #define _POSIX_READER_WRITER_LOCKS              200112L
161 #define _POSIX_SPAWN                            200112L
162 #define _POSIX_SPIN_LOCKS                       200112L
163 #define _POSIX_TIMEOUTS                         200112L

165 /*
166  * Support for the POSIX.1 mutex protocol attribute. For realtime applications
167  * which need mutexes to support priority inheritance/ceiling.
168  */
169 #if defined(_XPG6)
170 #define _POSIX_THREAD_PRIO_INHERIT              200112L
171 #define _POSIX_THREAD_PRIO_PROTECT              200112L
172 #else
173 #define _POSIX_THREAD_PRIO_INHERIT              1
174 #define _POSIX_THREAD_PRIO_PROTECT              1
175 #endif

177 #ifndef _POSIX_VDISABLE
178 #define _POSIX_VDISABLE         0
179 #endif

181 #ifndef NULL
182 #if defined(_LP64)
183 #define NULL    0L
184 #else
185 #define NULL    0
186 #endif
187 #endif

189 #define STDIN_FILENO    0
190 #define STDOUT_FILENO   1
191 #define STDERR_FILENO   2

193 /*
```



```
194  * Large File Summit-related announcement macros.  The system supports both
195  * the additional and transitional Large File Summit interfaces.  (The final
196  * two macros provide a finer granularity breakdown of _LFS64_LARGEFILE.)
197  */
198 #define _LFS_LARGEFILE          1
199 #define _LFS64_LARGEFILE        1
200 #define _LFS64_STDIO            1
201 #define _LFS64_ASYNCHRONOUS_IO  1

203 /* large file compilation environment setup */
204 #if !defined(_LP64) && _FILE_OFFSET_BITS == 64
205 #ifdef  __PRAGMA_REDEFINE_EXTNAME
206 #pragma redefine_extname        ftruncate       ftruncate64
207 #pragma redefine_extname        lseek           lseek64
208 #pragma redefine_extname        pread           pread64
209 #pragma redefine_extname        pwrite          pwrite64
210 #pragma redefine_extname        truncate        truncate64
211 #pragma redefine_extname        lockf           lockf64
212 #pragma redefine_extname        tell            tell64
213 #else   /* __PRAGMA_REDEFINE_EXTNAME */
214 #define ftruncate                       ftruncate64
215 #define lseek                           lseek64
216 #define pread                           pread64
217 #define pwrite                          pwrite64
218 #define truncate                        truncate64
219 #define lockf                           lockf64
220 #define tell                            tell64
221 #endif  /* __PRAGMA_REDEFINE_EXTNAME */
222 #endif  /* !_LP64 && _FILE_OFFSET_BITS == 64 */

224 /* In the LP64 compilation environment, the APIs are already large file */
225 #if defined(_LP64) && defined(_LARGEFILE64_SOURCE)
226 #ifdef  __PRAGMA_REDEFINE_EXTNAME
227 #pragma redefine_extname        ftruncate64     ftruncate
228 #pragma redefine_extname        lseek64         lseek
229 #pragma redefine_extname        pread64         pread
230 #pragma redefine_extname        pwrite64        pwrite
231 #pragma redefine_extname        truncate64      truncate
232 #pragma redefine_extname        lockf64         lockf
233 #pragma redefine_extname        tell64          tell
234 #else   /* __PRAGMA_REDEFINE_EXTNAME */
235 #define ftruncate64                     ftruncate
236 #define lseek64                         lseek
237 #define pread64                         pread
238 #define pwrite64                        pwrite
239 #define truncate64                      truncate
240 #define lockf64                         lockf
241 #define tell64                          tell
242 #endif  /* __PRAGMA_REDEFINE_EXTNAME */
243 #endif  /* _LP64 && _LARGEFILE64_SOURCE */

245 extern int access(const char *, int);
246 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
247 extern int acct(const char *);
248 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */
249 extern unsigned alarm(unsigned);
250 /* Marked as LEGACY in SUSv2 and removed in SUSv3 */
251 #if !defined(__XOPEN_OR_POSIX) || (defined(_XPG4_2) && !defined(_XPG6)) || \
252         defined(__EXTENSIONS__)
253 extern int brk(void *);
254 #endif /* !defined(__XOPEN_OR_POSIX) || (defined(_XPG4_2)... */
255 extern int chdir(const char *);
256 extern int chown(const char *, uid_t, gid_t);
257 /* Marked as LEGACY in SUSv2 and removed in SUSv3 */
258 #if !defined(_POSIX_C_SOURCE) || (defined(_XOPEN_SOURCE) && \
259         !defined(_XPG6)) || defined(__EXTENSIONS__)
```

```
260 extern int chroot(const char *);
261 #endif /* !defined(_POSIX_C_SOURCE) || defined(_XOPEN_SOURCE))... */
262 extern int close(int);
263 #if defined(_XPG4) || defined(__EXTENSIONS__)
264 extern size_t confstr(int, char *, size_t);
265 extern char *crypt(const char *, const char *);
266 #endif /* defined(_XPG4) || defined(__EXTENSIONS__) */
267 #if !defined(_POSIX_C_SOURCE) || defined(_XOPEN_SOURCE) || \
268         defined(__EXTENSIONS__)
269 extern char *ctermid(char *);
270 #endif /* (!defined(_POSIX_C_SOURCE) ... */
271 #if !defined(__XOPEN_OR_POSIX) || defined(_REENTRANT) || defined(__EXTENSIONS__)
272 extern char *ctermid_r(char *);
273 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_REENTRANT) ... */
274 /* Marked as LEGACY in SUSv2 and removed in SUSv3 */
275 #if !defined(_XPG6) || defined(__EXTENSIONS__)
276 extern char *cuserid(char *);
277 #endif
278 extern int dup(int);
279 extern int dup2(int, int);
280 extern int dup3(int, int, int);
281 #if defined(_XPG4) || defined(__EXTENSIONS__)
282 extern void encrypt(char *, int);
283 #endif /* defined(XPG4) || defined(__EXTENSIONS__) */
284 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
285 extern void endusershell(void);
286 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */
287 extern int execl(const char *, const char *, ...);
288 extern int execle(const char *, const char *, ...);
289 extern int execlp(const char *, const char *, ...);
290 extern int execv(const char *, char *const *);
291 extern int execve(const char *, char *const *, char *const *);
292 extern int execvp(const char *, char *const *);
293 extern void _exit(int)
294         __NORETURN;
295 /*
296  * The following fattach prototype is duplicated in <stropts.h>. The
297  * duplication is necessitated by XPG4.2 which requires the prototype
298  * be defined in <stropts.h>.
299  */
300 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
301 extern int fattach(int, const char *);
302 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */
303 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || defined(__EXTENSIONS__)
304 extern int fchdir(int);
305 extern int fchown(int, uid_t, gid_t);
306 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2)... */
307 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
308 extern int fchroot(int);
309 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */
310 #if !defined(__XOPEN_OR_POSIX) || (_POSIX_C_SOURCE > 2) || \
311         defined(__EXTENSIONS__)
312 extern int fdatasync(int);
313 #endif /* !defined(__XOPEN_OR_POSIX) || (_POSIX_C_SOURCE > 2)... */
314 /*
315  * The following fdetach prototype is duplicated in <stropts.h>. The
316  * duplication is necessitated by XPG4.2 which requires the prototype
317  * be defined in <stropts.h>.
318  */
319 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
320 extern int fdetach(const char *);
321 #endif /* !defined(__XOPEN_OR_POSIX)... */
322 extern pid_t fork(void);
323 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
324 extern pid_t fork1(void);
325 extern pid_t forkall(void);
```



```
326 #endif /* !defined(__XOPEN_OR_POSIX)... */
327 extern long fpathconf(int, int);
328 #if !defined(_POSIX_C_SOURCE) || (_POSIX_C_SOURCE > 2) || \
329         defined(__EXTENSIONS__)
330 extern int fsync(int);
331 #endif /* !defined(_POSIX_C_SOURCE) || (_POSIX_C_SOURCE > 2)... */
332 #if !defined(__XOPEN_OR_POSIX) || (_POSIX_C_SOURCE > 2) || defined(_XPG4_2) || \
333         (defined(_LARGEFILE_SOURCE) && _FILE_OFFSET_BITS == 64) || \
334         defined(__EXTENSIONS__)
335 extern int ftruncate(int, off_t);
336 #endif /* !defined(__XOPEN_OR_POSIX) || (_POSIX_C_SOURCE > 2)... */
337 extern char *getcwd(char *, size_t);
338 #if !defined(__XOPEN_OR_POSIX) || (defined(_XPG4_2) && !defined(_XPG6)) || \
339         defined(__EXTENSIONS__)
340 extern int getdtablesize(void);
341 #endif
342 extern gid_t getegid(void);
343 extern uid_t geteuid(void);
344 extern gid_t getgid(void);
345 extern int getgroups(int, gid_t *);
346 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || defined(__EXTENSIONS__)
347 extern long gethostid(void);
348 #endif
349 #if defined(_XPG4_2)
350 extern int gethostname(char *, size_t);
351 #elif  !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
352 extern int gethostname(char *, int);
353 #endif

355 #ifndef __GETLOGIN_DEFINED      /* Avoid duplicate in stdlib.h */
356 #define __GETLOGIN_DEFINED
357 #ifndef __USE_LEGACY_LOGNAME__
358 #ifdef  __PRAGMA_REDEFINE_EXTNAME
359 #pragma redefine_extname getlogin getloginx
360 #else   /* __PRAGMA_REDEFINE_EXTNAME */
361 extern char *getloginx(void);
362 #define getlogin        getloginx
363 #endif  /* __PRAGMA_REDEFINE_EXTNAME */
364 #endif  /* __USE_LEGACY_LOGNAME__ */
365 extern char *getlogin(void);
366 #endif  /* __GETLOGIN_DEFINED */

368 #if defined(_XPG4) || defined(__EXTENSIONS__)
369 extern int  getopt(int, char *const *, const char *);
370 extern char *optarg;
371 extern int  opterr, optind, optopt;
372 /* Marked as LEGACY in SUSv2 and removed in SUSv3 */
373 #if !defined(_XPG6) || defined(__EXTENSIONS__)
374 extern char *getpass(const char *);
375 #endif
376 #endif /* defined(_XPG4) || defined(__EXTENSIONS__) */
377 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || defined(__EXTENSIONS__)
378 /* Marked as LEGACY in SUSv2 and removed in SUSv3 */
379 #if !defined(_XPG6) || defined(__EXTENSIONS__)
380 extern int getpagesize(void);
381 #endif
382 extern pid_t getpgid(pid_t);
383 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2)... */
384 extern pid_t getpid(void);
385 extern pid_t getppid(void);
386 extern pid_t getpgrp(void);

388 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
389 char *gettxt(const char *, const char *);
390 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */
391 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || defined(__EXTENSIONS__)
```

```
392 extern pid_t getsid(pid_t);
393 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2)... */
394 extern uid_t getuid(void);
395 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
396 extern char *getusershell(void);
397 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */
398 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || defined(__EXTENSIONS__)
399 extern char *getwd(char *);
400 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2)... */
398 /*
399  * The following ioctl prototype is duplicated in <stropts.h>. The
400  * duplication is necessitated by XPG4.2 which requires the prototype
401  * be defined in <stropts.h>.
402  */
403 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
404 extern int ioctl(int, int, ...);
405 extern int isaexec(const char *, char *const *, char *const *);
406 extern int issetugid(void);
407 #endif
408 extern int isatty(int);
409 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || defined(__EXTENSIONS__)
410 extern int lchown(const char *, uid_t, gid_t);
411 #endif
412 extern int link(const char *, const char *);
413 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
414 extern offset_t llseek(int, offset_t, int);
415 #endif
416 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || \
417         (defined(_LARGEFILE_SOURCE) && _FILE_OFFSET_BITS == 64) || \
418         defined(__EXTENSIONS__)
419 extern int lockf(int, int, off_t);
420 #endif
421 extern off_t lseek(int, off_t, int);
422 #if !defined(_POSIX_C_SOURCE) || defined(_XOPEN_SOURCE) || \
423         defined(__EXTENSIONS__)
424 extern int nice(int);
425 #endif /* !defined(_POSIX_C_SOURCE) || defined(_XOPEN_SOURCE)... */
426 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
427 extern int mincore(caddr_t, size_t, char *);
428 #endif
429 extern long pathconf(const char *, int);
430 extern int pause(void);
431 extern int pipe(int *);
432 extern int pipe2(int *, int);
433 #if !defined(_POSIX_C_SOURCE) || defined(_XPG5) || \
434         (defined(_LARGEFILE_SOURCE) && _FILE_OFFSET_BITS == 64) || \
435         defined(__EXTENSIONS__)
436 extern ssize_t pread(int, void *, size_t, off_t);
437 #endif
438 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
439 extern void profil(unsigned short *, size_t, unsigned long, unsigned int);
440 #endif
441 /*
442  * pthread_atfork() is also declared in <pthread.h> as per SUSv3. The
443  * declarations are identical. A change to either one may also require
444  * appropriate namespace updates in order to avoid redeclaration
445  * warnings in the case where both prototypes are exposed via inclusion
446  * of both <pthread.h> and <unistd.h>.
447  */
448 #if !defined(__XOPEN_OR_POSIX) || \
449         ((_POSIX_C_SOURCE > 2) && !defined(_XPG6)) || \
450         defined(__EXTENSIONS__)
451 extern int pthread_atfork(void (*) (void), void (*) (void), void (*) (void));
452 #endif /* !defined(__XOPEN_OR_POSIX) || ((_POSIX_C_SOURCE > 2) ... */
453 #if !defined(_LP64) && \
454         (!defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__))
```



```
455 extern int ptrace(int, pid_t, int, int);
456 #endif
457 #if !defined(_POSIX_C_SOURCE) || defined(_XPG5) || \
458         (defined(_LARGEFILE_SOURCE) && _FILE_OFFSET_BITS == 64) || \
459         defined(__EXTENSIONS__)
460 extern ssize_t pwrite(int, const void *, size_t, off_t);
461 #endif
462 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
463 /* per RFC 3542; This is also defined in netdb.h */
464 extern int rcmd_af(char **, unsigned short, const char *, const char *,
465         const char *, int *, int);
466 #endif
467 extern ssize_t read(int, void *, size_t);
468 #if !defined(__XOPEN_OR_POSIX) || \
469         defined(_XPG4_2) || defined(__EXTENSIONS__)
470 extern ssize_t readlink(const char *_RESTRICT_KYWD, char *_RESTRICT_KYWD,
471         size_t);
472 #endif
473 #if (!defined(__XOPEN_OR_POSIX) || (defined(_XPG3) && !defined(_XPG4))) || \
474         defined(__EXTENSIONS__)
475 #if __cplusplus >= 199711L
476 namespace std {
477 #endif
478 extern int rename(const char *, const char *);
479 #if __cplusplus >= 199711L
480 } /* end of namespace std */

482 using std::rename;
483 #endif /* __cplusplus >= 199711L */
484 #endif /* (!defined(__XOPEN_OR_POSIX) || (defined(_XPG3)... */
485 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
486 extern int resolvepath(const char *, char *, size_t);
487 /* per RFC 3542; This is also defined in netdb.h */
488 extern int rexec_af(char **, unsigned short, const char *, const char *,
489         const char *, int *, int);
490 #endif /* !defined(__XOPEN_OR_POSIX)|| defined(__EXTENSIONS__) */
491 extern int rmdir(const char *);
492 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
493 /* per RFC 3542; This is also defined in netdb.h */
494 extern int rresvport_af(int *, int);
495 #endif

497 #if !defined(__XOPEN_OR_POSIX) || (defined(_XPG4_2) && !defined(_XPG6)) || \
498         defined(__EXTENSIONS__)
499 extern void *sbrk(intptr_t);
500 #endif /* !defined(__XOPEN_OR_POSIX) || (defined(_XPG4_2)... */
501 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG6) || defined(__EXTENSIONS__)
502 extern int setegid(gid_t);
503 extern int seteuid(uid_t);
504 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG6) ... */
505 extern int setgid(gid_t);
506 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
507 extern int setgroups(int, const gid_t *);
508 extern int sethostname(char *, int);
509 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */
510 extern int setpgid(pid_t, pid_t);
511 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || defined(__EXTENSIONS__)
512 extern pid_t setpgrp(void);
513 extern int setregid(gid_t, gid_t);
514 extern int setreuid(uid_t, uid_t);
515 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2)... */
516 extern pid_t setsid(void);
517 extern int setuid(uid_t);
518 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
519 extern void setusershell(void);
520 #endif /* !defined(__XOPEN_OR_POSIX)|| defined(__EXTENSIONS__) */
```

```
521 extern unsigned sleep(unsigned);
522 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
523 extern int stime(const time_t *);
524 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */
525 #if defined(_XPG4)
526 /* __EXTENSIONS__ makes the SVID Third Edition prototype in stdlib.h visible */
527 extern void swab(const void *_RESTRICT_KYWD, void *_RESTRICT_KYWD, ssize_t);
528 #endif /* defined(_XPG4) */
529 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || defined(__EXTENSIONS__)
530 extern int symlink(const char *, const char *);
531 extern void sync(void);
532 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) */
533 #if defined(_XPG5) && !defined(_XPG6)
534 #ifdef __PRAGMA_REDEFINE_EXTNAME
535 #pragma redefine_extname sysconf __sysconf_xpg5
536 #else /* __PRAGMA_REDEFINE_EXTNAME */
537 #define sysconf __sysconf_xpg5
538 #endif  /* __PRAGMA_REDEFINE_EXTNAME */
539 #endif /* defined(_XPG5) && !defined(_XPG6) */
540 extern long sysconf(int);
541 extern pid_t tcgetpgrp(int);
542 extern int tcsetpgrp(int, pid_t);
543 #if !defined(__XOPEN_OR_POSIX) || \
544         (defined(_LARGEFILE_SOURCE) && _FILE_OFFSET_BITS == 64) || \
545         defined(__EXTENSIONS__)
546 extern off_t tell(int);
547 #endif /* !defined(__XOPEN_OR_POSIX)... */
548 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || \
549         (defined(_LARGEFILE_SOURCE) && _FILE_OFFSET_BITS == 64) || \
550         defined(__EXTENSIONS__)
551 extern int truncate(const char *, off_t);
552 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2)... */
553 extern char *ttyname(int);
554 #if (defined(_XPG4_2) && !defined(_XPG7)) || !defined(_STRICT_SYMBOLS)
555 extern useconds_t ualarm(useconds_t, useconds_t);
556 #endif
557 extern int unlink(const char *);
558 #if (defined(_XPG4_2) && !defined(_XPG7)) || !defined(_STRICT_SYMBOLS)
559 extern char *getwd(char *);
560 extern int usleep(useconds_t);
561 extern pid_t vfork(void) __RETURNS_TWICE;
562 #pragma unknown_control_flow(vfork)
563 #endif
564 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || defined(__EXTENSIONS__)
565 extern pid_t vfork(void) __RETURNS_TWICE;
566 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2)... */
564 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
565 extern void vhangup(void);
566 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */
567 extern ssize_t write(int, const void *, size_t);
568 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
569 extern void yield(void);
570 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */

572 #if !defined(__XOPEN_OR_POSIX) || defined(_ATFILE_SOURCE) || \
573         defined(__EXTENSIONS__)
574         /* || defined(_XPG7) */
575 extern int faccessat(int, const char *, int, int);
576 extern int fchownat(int, const char *, uid_t, gid_t, int);
577 extern int linkat(int, const char *, int, const char *, int);
578 extern ssize_t readlinkat(int, const char *_RESTRICT_KYWD,
579         char *_RESTRICT_KYWD, size_t);
580 extern int renameat(int, const char *, int, const char *);
581 extern int symlinkat(const char *, int, const char *);
582 extern int unlinkat(int, const char *, int);
583 #endif  /* !defined(__XOPEN_OR_POSIX) || defined(_ATFILE_SOURCE)... */
```



```
584 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
585 extern int get_nprocs(void);
586 extern int get_nprocs_conf(void);
587 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */

589 /* transitional large file interface versions */
590 #if     defined(_LARGEFILE64_SOURCE) && !((_FILE_OFFSET_BITS == 64) && \
591             !defined(__PRAGMA_REDEFINE_EXTNAME))
592 extern int ftruncate64(int, off64_t);
593 extern off64_t lseek64(int, off64_t, int);
594 extern ssize_t  pread64(int, void *, size_t, off64_t);
595 extern ssize_t  pwrite64(int, const void *, size_t, off64_t);
596 extern off64_t  tell64(int);
597 extern int      truncate64(const char *, off64_t);
598 extern int      lockf64(int, int, off64_t);
599 #endif  /* _LARGEFILE64_SOURCE */

604 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) || defined(__EXTENSIONS__)
605 #pragma unknown_control_flow(vfork)
606 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2)... */

601 /*
602  * getlogin_r() & ttyname_r() prototypes are defined here.
603  */

605 /*
606  * Previous releases of Solaris, starting at 2.3, provided definitions of
607  * various functions as specified in POSIX.1c, Draft 6.  For some of these
608  * functions, the final POSIX 1003.1c standard had a different number of
609  * arguments and return values.
610  *
611  * The following segment of this header provides support for the standard
612  * interfaces while supporting applications written under earlier
613  * releases.  The application defines appropriate values of the feature
614  * test macros _POSIX_C_SOURCE and _POSIX_PTHREAD_SEMANTICS to indicate
615  * whether it was written to expect the Draft 6 or standard versions of
616  * these interfaces, before including this header.  This header then
617  * provides a mapping from the source version of the interface to an
618  * appropriate binary interface.  Such mappings permit an application
619  * to be built from libraries and objects which have mixed expectations
620  * of the definitions of these functions.
621  *
622  * For applications using the Draft 6 definitions, the binary symbol is the
623  * same as the source symbol, and no explicit mapping is needed.  For the
624  * standard interface, the function func() is mapped to the binary symbol
625  * _posix_func().  The preferred mechanism for the remapping is a compiler
626  * #pragma.  If the compiler does not provide such a #pragma, the header file
627  * defines a static function func() which calls the _posix_func() version;
628  * this has to be done instead of #define since POSIX specifies that an
629  * application can #undef the symbol and still be bound to the correct
630  * implementation.  Unfortunately, the statics confuse lint so we fallback to
631  * #define in that case.
632  *
633  * NOTE: Support for the Draft 6 definitions is provided for compatibility
634  * only.  New applications/libraries should use the standard definitions.
635  */

637 #if     defined(__EXTENSIONS__) || defined(_REENTRANT) || \
638         !defined(__XOPEN_OR_POSIX) || (_POSIX_C_SOURCE - 0 >= 199506L) || \
639         defined(_POSIX_PTHREAD_SEMANTICS)

641 #if     (_POSIX_C_SOURCE - 0 >= 199506L) || defined(_POSIX_PTHREAD_SEMANTICS)

643 #ifndef __USE_LEGACY_LOGNAME__
644 #ifdef  __PRAGMA_REDEFINE_EXTNAME
645 #pragma redefine_extname getlogin_r __posix_getloginx_r
```

```
 646 extern int getlogin_r(char *, int);
 647 #else   /* __PRAGMA_REDEFINE_EXTNAME */
 648 extern int __posix_getloginx_r(char *, int);
 649 #define getlogin_r      __posix_getloginx_r
 650 #endif  /* __PRAGMA_REDEFINE_EXTNAME */
 651 #else   /* __USE_LEGACY_LOGNAME__ */
 652 #ifdef __PRAGMA_REDEFINE_EXTNAME
 653 #pragma redefine_extname getlogin_r __posix_getlogin_r
 654 extern int getlogin_r(char *, int);
 655 #else  /* __PRAGMA_REDEFINE_EXTNAME */
 656 extern int __posix_getlogin_r(char *, int);

 658 #ifdef __lint

 660 #define getlogin_r      __posix_getlogin_r

 662 #else /* !__lint */

 664 static int
 665 getlogin_r(char *__name, int __len)
 666 {
 667         return (__posix_getlogin_r(__name, __len));
 668 }
```

unchanged_portion_omitted_

```
**********************************************************
    5907 Sun Mar 29 19:44:28 2015
new/usr/src/man/man2/vfork.2
5776 vfork and getwd should not be exposed under XPG7
**********************************************************
```

```
.\" Copyright 2014 Garrett D'Amore <garrett@damore.org>
'\" te
.\" Copyright (c) 2004, Sun Microsystems, Inc.  All Rights Reserved.
.\" Copyright 1989 AT&T.
.\" Copyright (c) 1980 Regents of the University of California.
.\" All rights reserved.  The Berkeley software License Agreement
.\" specifies the terms and conditions for redistribution.
.Dd Aug 20, 2014
.Dt VFORK 2
.Os
.Sh NAME
.Nm vfork ,
.Nm vforkx
.Nd spawn new process in a virtual memory efficient way
.Sh SYNOPSIS
.In unistd.h
.Ft pid_t
.Fn vfork void
.
.In sys/fork.h
.Ft pid_t
.Fn vforkx "int flags"
.Sh DESCRIPTION
The
.Fn vfork
and
.Fn vforkx
functions create a new process without
```

```
.\" Copyright (c) 1980 Regents of the University of California.  All rights rese
.TH VFORK 2 "Dec 13, 2006"
.SH NAME
vfork, vforkx \- spawn new process in a virtual memory efficient way
.SH SYNOPSIS
.LP
.nf
#include <unistd.h>

\fBpid_t\fR \fBvfork\fR(\fBvoid\fR);
.fi

.LP
.nf
#include <sys/fork.h>

\fBpid_t\fR \fBvforkx\fR(\fBint\fR \fIflags\fR);
.fi

.SH DESCRIPTION
.sp
.LP
The \fBvfork()\fR and \fBvforkx()\fR functions create a new process without
```

```
fully copying the address space of the old process. These functions are useful
in instances where the purpose of a
.Xr fork 2
operation is to create a new
system context for an
.Xr exec 2
operation.
.Lp
Unlike with the
.Fn fork
```



```
function, the child process borrows the parent's
memory and thread of control until a call to
.Fn execve
or an exit
.Pq either abnormally or by a call to Xr _exit 2 .
Any modification
```

```
in instances where the purpose of a \fBfork\fR(2) operation is to create a new
system context for an \fBexecve()\fR operation (see \fBexec\fR(2)).
.sp
.LP
Unlike with the \fBfork()\fR function, the child process borrows the parent's
memory and thread of control until a call to \fBexecve()\fR or an exit (either
abnormally or by a call to \fB_exit()\fR (see \fBexit\fR(2)). Any modification
```

```
made during this time to any part of memory in the child process is reflected
in the parent process on return from
.Fn vfork
or
.Fn vforkx .
The parent process is suspended while the child is using its resources.
.Lp
In a multithreaded application,
.Fn vfork
and
.Fn vforkx
borrow only the thread of control that called
.Fn vfork
or
.Fn vforkx
in the parent; that is, the child contains only one thread. The use of
.Fn vfork
or
.Fn vforkx
in multithreaded applications, however, is unsafe due to race
```

```
in the parent process on return from \fBvfork()\fR or \fBvforkx()\fR. The
parent process is suspended while the child is using its resources.
.sp
.LP
In a multithreaded application,  \fBvfork()\fR and \fBvforkx()\fR borrow only
the thread of control that called \fBvfork()\fR or \fBvforkx()\fR in the
parent; that is, the child contains only one thread. The use of \fBvfork()\fR
or \fBvforkx()\fR in multithreaded applications, however, is unsafe due to race
```

```
conditions that can cause the child process to become deadlocked and
consequently block both the child and parent process from execution
indefinitely.
.Lp
The
.Fn vfork
and
.Fn vforkx
functions can normally be used the same way as
.Fn fork
and
.Fn forkx ,
respectively. The calling procedure,
```

```
.sp
.LP
The \fBvfork()\fR and \fBvforkx()\fR functions can normally be used the same
way as \fBfork()\fR and \fBforkx()\fR, respectively. The calling procedure,
```

```
however, should not return while running in the child's context, since the
eventual return from
.Fn vfork
or
.Fn vforkx
in the parent would be to
a stack frame that no longer exists. The
.Fn _exit
```

```
 85 function should be used
 86 in favor of
 87 .Xr exit 3C
 88 if unable to perform an
 89 .Fn execve
 90 operation, since
 91 .Fn exit
 92 will invoke all functions registered by
 93 .Xr atexit 3C
 94 and will flush and close standard I/O channels, thereby corrupting the parent
 52 eventual return from \fBvfork()\fR or \fBvforkx()\fR in the parent would be to
 53 a stack frame that no longer exists. The \fB_exit()\fR function should be used
 54 in favor of \fBexit\fR(3C) if unable to perform an \fBexecve()\fR operation,
 55 since \fBexit()\fR will invoke all functions registered by \fBatexit\fR(3C) and
 56 will flush and close standard I/O channels, thereby corrupting the parent
 95 process's standard I/O data structures. Care must be taken in the child process
 96 not to modify any global or local data that affects the behavior of the parent
 97 process on return from
 98 .Fn vfork
 99 or
100 .Fn vforkx ,
101 unless such an effect
 59 process on return from \fBvfork()\fR or \fBvforkx()\fR, unless such an effect
102 is intentional.
103 .Lp
104 Unlike
105 .Fn fork
106 and
107 .Fn forkx ,
108 fork handlers are not run when
109 .Fn vfork
110 and
111 .Fn vforkx
112 are called.
113 .Lp
114 The
115 .Fn vfork
116 and
117 .Fn vforkx
118 functions are deprecated. Their sole
 61 .sp
 62 .LP
 63 Unlike \fBfork()\fR and \fBforkx()\fR, fork handlers are not run when
 64 \fBvfork()\fR and \fBvforkx()\fR are called.
 65 .sp
 66 .LP
 67 The \fBvfork()\fR and \fBvforkx()\fR functions are deprecated. Their sole
119 legitimate use as a prelude to an immediate call to a function from the
120 .Xr exec 2
121 family can be achieved safely by
122 .Xr posix_spawn 3C
123 or
124 .Xr posix_spawnp 3C .
125 .Ss "Fork Extensions"
126 The
127 .Fn vforkx
128 function accepts a
129 .Fa flags
130 argument consisting of a
 69 \fBexec\fR family can be achieved safely by \fBposix_spawn\fR(3C) or
 70 \fBposix_spawnp\fR(3C).
 71 .SS "Fork Extensions"
 72 .sp
 73 .LP
 74 The \fBvforkx()\fR function accepts a \fIflags\fR argument consisting of a
131 bitwise inclusive-OR of zero or more of the following flags, which are defined
```



```
132 in the header
133 .In sys/fork.h :
134 .Lp
135 .Bl -item -compact -offset indent
136 .It
137 .Dv FORK_NOSIGCHLD
138 .It
139 .Dv FORK_WAITPID
140 .El
141 .Lp
142 See
143 .Xr fork 2
144 for descriptions of these flags. If the
145 .Fa flags
146 argument is 0,
147 .Fn vforkx
148 is identical to
149 .Fn vfork .
150 .Sh RETURN VALUES
151 Upon successful completion,
152 .Fn vfork
153 and
154 .Fn vforkx
155 return 0 to
156 the child process and return the process ID of the child process to the parent
157 process. Otherwise, \(mi1 is returned to the parent process, no child
158 process is created, and
159 .Va errno
160 is set to indicate the error.
161 .Sh ERRORS
162 The
163 .Fn vfork
164 and
165 .Fn vforkx
166 functions will fail if:
167 .Bl -tag -width Er
168 .It Er EAGAIN
 76 in the header \fB<sys/fork.h>\fR:
 77 .br
 78 .in +2
 79 \fBFORK_NOSIGCHLD\fR
 80 .in -2
 81 .br
 82 .in +2
 83 \fBFORK_WAITPID\fR
 84 .in -2
 85 .sp
 86 .LP
 87 See \fBfork\fR(2) for descriptions of these flags. If the \fIflags\fR argument
 88 is 0, \fBvforkx()\fR is identical to \fBvfork()\fR.
 89 .SH RETURN VALUES
 90 .sp
 91 .LP
 92 Upon successful completion, \fBvfork()\fR and \fBvforkx()\fR return  \fB0\fR to
 93 the child process and returns the process ID of the child process to the parent
 94 process. Otherwise, \fB\(mi1\fR is returned to the parent process, no child
 95 process is created, and \fBerrno\fR is set to indicate the error.
 96 .SH ERRORS
 97 .sp
 98 .LP
 99 The \fBvfork()\fR and \fBvforkx()\fR functions will fail if:
100 .sp
101 .ne 2
102 .na
103 \fB\fBEAGAIN\fR\fR
104 .ad
```

```
.RS 10n
The system-imposed limit on the total number of processes under execution
(either system-quality or by a single user) would be exceeded. This limit is
determined when the system is generated.
.
.It Er ENOMEM
.RE

.sp
.ne 2
.na
\fB\fBENOMEM\fR\fR
.ad
.RS 10n
There is insufficient swap space for the new process.
.El
.Lp
The
.Fn vforkx
function will fail if:
.Bl -tag -width Er
.It Er EINVAL
The
.Va flags
argument is invalid.
.El
.Sh INTERFACE STABILITY
The
.Fn vfork
function is
.Sy Obsolete Standard .
.Lp
The
.Fn vforkx
function is
.Sy Obsolete Uncommitted .
.Sh MT-LEVEL
.Sy Unsafe .
.Sh SEE ALSO
.Xr exec 2 ,
.Xr exit 2 ,
.Xr fork 2 ,
.Xr ioctl 2 ,
.Xr atexit 3C ,
.Xr exit 3C ,
.Xr posix_spawn 3C ,
.Xr posix_spawnp 3C ,
.Xr signal.h 3HEAD ,
.Xr wait 3C ,
.Xr standards 5
.Sh NOTES
.RE

.sp
.LP
The \fBvforkx()\fR function will fail if:
.sp
.ne 2
.na
\fB\fBEINVAL\fR\fR
.ad
.RS 10n
The \fIflags\fR argument is invalid.
.RE

.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     Obsolete
_
MT-Level        Unsafe
.TE

.SH SEE ALSO
.sp
.LP
\fBexec\fR(2), \fBexit\fR(2), \fBfork\fR(2), \fBioctl\fR(2), \fBatexit\fR(3C),
\fBexit\fR(3C), \fBposix_spawn\fR(3C), \fBposix_spawnp\fR(3C),
\fBsignal.h\fR(3HEAD), \fBwait\fR(3C), \fBattributes\fR(5), \fBstandards\fR(5)
.SH NOTES
.sp
.LP
To avoid a possible deadlock situation, processes that are children in the
middle of a
.Fn vfork
or
.Fn vforkx
are never sent
.Dv SIGTTOU
or
.Dv SIGTTIN
signals; rather, output or ioctls are allowed and input attempts
result in an
.Dv EOF
indication.
.Lp
middle of a \fBvfork()\fR or \fBvforkx()\fR are never sent \fBSIGTTOU\fR or
\fBSIGTTIN\fR signals; rather, output or ioctls are allowed and input attempts
result in an \fBEOF\fR indication.
.sp
.LP
To forestall parent memory corruption due to race conditions with signal
handling,
.Fn vfork
and
.Fn vforkx
treat signal handlers in the child
process in the same manner as the
.Xr exec 2
functions: signals set to be
caught by the parent process are set to the default action
.Pq Dv SIG_DFL
in the child process
.Pq see Xr signal.h 3HEAD .
Any attempt to set a signal
handler in the child before
.Fn execve
to anything other than
.Dv SIG_DFL
or
.Dv SIG_IGN
is disallowed and results in setting the handler to
```

```
.Dv SIG_DFL .
.Lp
On some systems, the implementation of
.Fn vfork
and
.Fn vforkx
cause
handling, \fBvfork()\fR and \fBvforkx()\fR treat signal handlers in the child
process in the same manner as the \fBexec\fR(2) functions: signals set to be
caught by the parent process are set to the default action (\fBSIG_DFL\fR) in
the child process (see \fBsignal.h\fR(3HEAD)). Any attempt to set a signal
handler in the child before \fBexecve()\fR to anything other than \fBSIG_DFL\fR
or \fBSIG_IGN\fR is disallowed and results in setting the handler to
\fBSIG_DFL\fR.
.sp
.LP
On some systems, the implementation of \fBvfork()\fR and \fBvforkx()\fR cause
```

the parent to inherit register values from the child. This can create problems

```
for certain optimizing compilers if
.In unistd.h
is not included in the source calling
.Fn vfork
or if
.In sys/fork.h
is not included in the
source calling
.Fn vforkx .
.Sh STANDARDS
The
.Fn vfork
function is available in the following compilation environments.  See
.Xr standards 5 .
.Lp
.Bl -bullet -compact
.It
.St -xpg4.2
.It
.St -susv2
.It
.St -susv3
.El
.Lp
It was marked obsolete in
.St -susv3
and removed from
.St -p1003.1-2008 .
.Lp
The
.Fn vforkx
function is a local extension and not available in any strictly
standards-compliant compilation environment.
for certain optimizing compilers if \fB<unistd.h>\fR is not included in the
source calling \fBvfork()\fR or if \fB<sys/fork.h>\fR is not included in the
source calling \fBvforkx()\fR.
```

```
**********************************************************
    3465 Sun Mar 29 19:44:32 2015
new/usr/src/man/man3c/getwd.3c
5776 vfork and getwd should not be exposed under XPG7
**********************************************************
```

```
.\" Copyright 2015 Garrett D'Amore <garrett@damore.org>
'\" te
.\" Copyright (c) 1992, X/Open Company Limited All Rights Reserved  Portions Cop
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" http://www.opengroup.org/bookstore/.
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\"  This notice shall appear on any product containing this material.
.\" 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
.Dd Mar 30, 2015
.Dt GETWD 3C
.Os
.Sh NAME
.Nm getwd
.Nd get current working directory pathname
.Sh SYNOPSIS
.In unistd.h
.Ft "char *"
.Fn getwd "char *path_name"
.Sh DESCRIPTION
The
.Fn getwd
function determines an absolute pathname of the current
.TH GETWD 3C "Jul 24, 2002"
.SH NAME
getwd \- get current working directory pathname
.SH SYNOPSIS
.LP
.nf
#include <unistd.h>

\fBchar *\fR\fBgetwd\fR(\fBchar *\fR\fIpath_name\fR);
.fi

.SH DESCRIPTION
.sp
.LP
The \fBgetwd()\fR function determines an absolute pathname of the current
working directory of the calling process, and copies that pathname into the
array pointed to by the
.Fa path_name
argument.
.Lp
array pointed to by the \fIpath_name\fR argument.
.sp
.LP
If the length of the pathname of the current working directory is greater than
.Pq Dv PATH_MAX + 1
including the null byte,
.Fn getwd
fails and returns a null pointer.
.Sh RETURN VALUES
(\fIPATH_MAX\fR + 1) including the null byte, \fBgetwd()\fR fails and returns a
null pointer.
.SH RETURN VALUES
.sp
.LP
Upon successful completion, a pointer to the string containing the absolute
pathname of the current working directory is returned.  Otherwise,
.Fn getwd
```



```
returns a null pointer and the contents of the array pointed to by
.Fa path_name
are undefined.
.Sh ERRORS
\fBgetwd()\fR returns a null pointer and the contents of the array pointed to
by \fIpath_name\fR are undefined.
.SH ERRORS
.sp
.LP
No errors are defined.
.Sh USAGE
The
.Fn getwd
function is supplied for backwards compatibility.  The
.Xr getcwd 3C
should be used instead.
.Sh INTERFACE STABILITY
.Sy Obsolete Standard .
.Sh SEE ALSO
.Xr getcwd 3C ,
.Xr standards 5
.Sh STANDARDS
The
.Fn getwd
function is available in the following compilation environments. See
.Xr standards 5 .
.Lp
.Bl -bullet -compact
.It
.St -xpg4.2
.It
.St -susv2
.It
.St -susv3
.El
.Lp
It was marked obsolete in
.St -susv3
and removed from
.St -p1003.1-2008 .
.SH USAGE
.sp
.LP
For portability to implementations conforming to versions of the X/Open
Portability Guide prior to SUS, \fBgetcwd\fR(3C) is preferred over this
function.
.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     Standard
.TE

.SH SEE ALSO
.sp
.LP
\fBgetcwd\fR(3C), \fBattributes\fR(5), \fBstandards\fR(5)
```

```
**********************************************************
    3933 Sun Mar 29 19:44:36 2015
new/usr/src/test/libc-tests/cfg/symbols/unistd_h.cfg
5776 vfork and getwd should not be exposed under XPG7
**********************************************************
```

```
#
# 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 2015 Garrett D'Amore <garrett@damore.org>
#

#
# Definitions found in unistd.h
#

#
# Types.
#
type | pid_t | unistd.h | POSIX+ SUS+

#
# Values.
#
# Note that the standard requires the user declare environ.
# value | environ | char ** | unistd.h | POSIX+ SUS+
value | _CS_PATH                                | int | unistd.h | SUS+

value | _CS_POSIX_V6_ILP32_OFF32_CFLAGS         | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_ILP32_OFF32_LDFLAGS        | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_ILP32_OFF32_LIBS           | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_ILP32_OFFBIG_CFLAGS        | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_ILP32_OFFBIG_LDFLAGS       | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_ILP32_OFFBIG_LIBS          | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_LP64_OFF64_CFLAGS          | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_LP64_OFF64_LDFLAGS         | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_LP64_OFF64_LIBS            | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_LPBIG_OFFBIG_CFLAGS        | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_LPBIG_OFFBIG_LDFLAGS       | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_LPBIG_OFFBIG_LIBS          | int | unistd.h | SUSv3+
value | _CS_POSIX_V6_WIDTH_RESTRICTED_ENVS      | int | unistd.h | SUSv3+

#
# Functions
#
func |  access                                                  |\
        int                                                     |\
        const char *; int                                       |\
        unistd.h | POSIX+ SUS+

func |  chown                                                   |\
        int                                                     |\
        const char *; uid_t; gid_t                              |\
        unistd.h | POSIX+ SUS+

func |  execl                                                   |\
        int                                                     |\
        const char *; const char *                              |\
```



```
        unistd.h | POSIX+ SUS+

func |  execle                                                  |\
        int                                                     |\
        const char *; const char *; char *; char *const []      |\
        unistd.h | POSIX+ SUS+

func |  execlp                                                  |\
        int                                                     |\
        const char *; const char *; char *                      |\
        unistd.h | POSIX+ SUS+

func |  execv                                                   |\
        int                                                     |\
        const char *; char *const []                            |\
        unistd.h | POSIX+ SUS+

func |  execve                                                  |\
        int                                                     |\
        const char *; char *const []; char *const []            |\
        unistd.h | POSIX+ SUS+

func |  execvp                                                  |\
        int                                                     |\
        const char *; char *const []                            |\
        unistd.h | POSIX+ SUS+

func |  fchown                                                  |\
        int                                                     |\
        int; uid_t; gid_t                                       |\
        unistd.h | -POSIX+ SUS+

func |  getlogin                                                |\
        char *                                                  |\
        void                                                    |\
        unistd.h | POSIX+ SUS+

func |  getlogin_r                                              |\
        int                                                     |\
        char *; size_t                                          |\
        unistd.h | -POSIX+ -SUS+ +POSIX-1995+ SUSv2+

func |  getwd                                                   |\
        char *                                                  |\
        char *                                                  |\
        unistd.h | -ALL SUS+ -SUSv4+

func |  lchown                                                  |\
        int                                                     |\
        const char *; uid_t; gid_t                              |\
        unistd.h | -POSIX+ SUS+

func |  link                                                    |\
        int                                                     |\
        const char *; const char *                              |\
        unistd.h | POSIX+ SUS+

# XPG3 may have put this here incorrectly (Open Group says no..., but...)
# Probably this is actually our error, and we should kill it, but we can
# do that when kill off XPG3 support altogether.
func |  rename                                                  |\
        int                                                     |\
        const char *; const char *                              |\
        unistd.h | -POSIX+ +XPG3 -XPG4+

func |  symlink                                                 |\
```

```
128         int                                                     |\
129         const char *; const char *                              |\
130         unistd.h | -XPG3+ -POSIX+ SUS+

132 func |  ttyname                                                  |\
133         char *                                                  |\
134         int                                                     |\
135         unistd.h | POSIX+ SUS+

137 func |  ttyname_r                                                |\
138         int                                                     |\
139         int; char *; size_t                                     |\
140         unistd.h | -POSIX+ -SUS+ +POSIX-1995+ SUSv2+

142 func |  ualarm                                                  |\
143         int                                                     |\
144         useconds_t; useconds_t                                        |\
145         unistd.h | -ALL SUS+ -SUSv4+

147 func |  unlink                                                  |\
148         int                                                     |\
149         const char *                                            |\
150         unistd.h | POSIX+ XPG3+

152 func |  usleep                                                  |\
153         int                                                     |\
154         useconds_t                                              |\
155         unistd.h | -ALL SUS+ -SUSv4+

157 func |  vfork                                                   |\
158         pid_t                                                   |\
159         void                                                    |\
160         unistd.h | -ALL SUS+ -SUSv4+
```