```
**********************************************************
   27029 Mon Jan 28 14:19:12 2013
new/usr/src/cmd/ssh/include/config.h
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
```

```
/* config.h.  Generated by configure.  */
/* config.h.in.  Generated from configure.ac by autoheader.  */
/* $Id: acconfig.h,v 1.145 2002/09/26 00:38:48 tim Exp $ */

/*
 * Copyright (c) 2001, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2012 Gary Mills
 */

#ifndef _CONFIG_H
#define _CONFIG_H

#ifdef __cplusplus
extern "C" {
#endif


/* Generated automatically from acconfig.h by autoheader. */
/* Please make your changes there */


/* Define to a Set Process Title type if your system is */
/* supported by bsd-setproctitle.c */
/* #undef SPT_TYPE */

/* setgroups() NOOP allowed */
/* #undef SETGROUPS_NOOP */

/* SCO workaround */
/* #undef BROKEN_SYS_TERMIO_H */

/* If your header files don't define LOGIN_PROGRAM, then use this (detected) */
/* from environment and PATH */
#define LOGIN_PROGRAM_FALLBACK "/usr/bin/login"

/* Define if your password has a pw_class field */
/* #undef HAVE_PW_CLASS_IN_PASSWD */

/* Define if your password has a pw_expire field */
/* #undef HAVE_PW_EXPIRE_IN_PASSWD */

/* Define if your password has a pw_change field */
/* #undef HAVE_PW_CHANGE_IN_PASSWD */

/* Define if your system uses access rights style file descriptor passing */
#define HAVE_ACCRIGHTS_IN_MSGHDR 1

/* Define if your system uses ancillary data style file descriptor passing */
/* #undef HAVE_CONTROL_IN_MSGHDR */

/* Define if you system's inet_ntoa is busted (e.g. Irix gcc issue) */
/* #undef BROKEN_INET_NTOA */

/* Define if your system defines sys_errlist[] */
#define HAVE_SYS_ERRLIST 1

/* Define if your system defines sys_nerr */
#define HAVE_SYS_NERR 1

/* Define if your system choked on IP TOS setting */
```



```
#define IP_TOS_IS_BROKEN 1

/* Define if you have the getuserattr function.  */
/* #undef HAVE_GETUSERATTR */

/* Work around problematic Linux PAM modules handling of PAM_TTY */
#define PAM_TTY_KLUDGE 1

/* Define if your snprintf is busted */
/* #undef BROKEN_SNPRINTF */

/* Define if you are on Cygwin */
/* #undef HAVE_CYGWIN */

/* Define if you have a broken realpath. */
/* #undef BROKEN_REALPATH */

/* Define if you are on NEWS-OS */
/* #undef HAVE_NEWS4 */

/* Define if you want to enable PAM support */
#define USE_PAM 1

/* Define if you want to enable AIX4's authenticate function */
/* #undef WITH_AIXAUTHENTICATE */

/*
 * Define if you have/want arrays (cluster-wide session managment, not C
 * arrays)
 */
/* #undef WITH_IRIX_ARRAY */

/* Define if you want IRIX project management */
/* #undef WITH_IRIX_PROJECT */

/* Define if you want IRIX audit trails */
/* #undef WITH_IRIX_AUDIT */

/* Define if you want IRIX kernel jobs */
/* #undef WITH_IRIX_JOBS */

/* Location of PRNGD/EGD random number socket */
/* #undef PRNGD_SOCKET */

/* Port number of PRNGD/EGD random number socket */
/* #undef PRNGD_PORT */

/* Builtin PRNG command timeout */
#define ENTROPY_TIMEOUT_MSEC 200

/* non-privileged user for privilege separation */
#define SSH_PRIVSEP_USER "sshd"

/* Define if you want to install preformatted manpages. */
/* #undef MANTYPE */

/* Define if your ssl headers are included with #include <openssl/header.h>  */
#define HAVE_OPENSSL 1

/* Define if Solaris' OpenSSL lacks AES support */
#define SOLARIS_OPENSSL_NO_AES 1

/* Define if Solaris-style Least Privilege is available */
#define HAVE_SOLARIS_PRIVILEGE 1

/* Define if you want Sun's alternative privilege separation */
```

```
127 #define ALTPRIVSEP

129 /* Define if you have Solaris-style Contracts */
130 #define HAVE_SOLARIS_CONTRACTS 1

132 /* Define if SVR4-style libcmd (for accessing /etc/default/ files) */
133 #define HAVE_DEFOPEN 1

135 /*
136  * Define if you are linking against RSAref. Used only to print the right
137  * message at run-time.
138  */
139 /* #undef RSAREF */

141 /* struct timeval */
142 #define HAVE_STRUCT_TIMEVAL 1

144 /* struct utmp and struct utmpx fields */
145 /* #undef HAVE_HOST_IN_UTMP */
146 #define HAVE_HOST_IN_UTMPX 1
147 /* #undef HAVE_ADDR_IN_UTMP */
148 /* #undef HAVE_ADDR_IN_UTMPX */
149 /* #undef HAVE_ADDR_V6_IN_UTMP */
150 /* #undef HAVE_ADDR_V6_IN_UTMPX */
151 #define HAVE_SYSLEN_IN_UTMPX 1
152 #define HAVE_PID_IN_UTMP 1
153 #define HAVE_TYPE_IN_UTMP 1
154 #define HAVE_TYPE_IN_UTMPX 1
155 /* #undef HAVE_TV_IN_UTMP */
156 #define HAVE_TV_IN_UTMPX 1
157 #define HAVE_ID_IN_UTMP 1
158 #define HAVE_ID_IN_UTMPX 1
159 #define HAVE_EXIT_IN_UTMP 1
160 #define HAVE_TIME_IN_UTMP 1
161 #define HAVE_TIME_IN_UTMPX 1

163 /* Define if you don't want to use your system's login() call */
164 /* #undef DISABLE_LOGIN */

166 /* Define if you don't want to use pututline() etc. to write [uw]tmp */
167 /* #undef DISABLE_PUTUTLINE */

169 /* Define if you don't want to use pututxline() etc. to write [uw]tmpx */
170 /* #undef DISABLE_PUTUTXLINE */

172 /* Define if you don't want to use lastlog */
173 /* #undef DISABLE_LASTLOG */

175 /* Define if you don't want to use lastlog in session.c */
176 /* #undef NO_SSH_LASTLOG */

178 /* Define if you don't want to use utmp */
179 #define DISABLE_UTMP 1

181 /* Define if you don't want to use utmpx */
182 /* #undef DISABLE_UTMPX */

184 /* Define if you don't want to use wtmp */
185 #define DISABLE_WTMP 1

187 /* Define if you don't want to use wtmpx */
188 /* #undef DISABLE_WTMPX */

190 /* Some systems need a utmpx entry for /bin/login to work */
191 #define LOGIN_NEEDS_UTMPX 1
```



```
193 /* Some versions of /bin/login need the TERM supplied on the commandline */
194 #define LOGIN_NEEDS_TERM 1

196 /* Define if your login program cannot handle end of options ("--") */
197 /* #undef LOGIN_NO_ENDOPT */

199 /* Define if you want to specify the path to your lastlog file */
200 #define CONF_LASTLOG_FILE "/var/adm/lastlog"

202 /* Define if you want to specify the path to your utmp file */
203 /* #undef CONF_UTMP_FILE */

205 /* Define if you want to specify the path to your wtmp file */
206 /* #undef CONF_WTMP_FILE */

208 /* Define if you want to specify the path to your utmpx file */
209 /* #undef CONF_UTMPX_FILE */

211 /* Define if you want to specify the path to your wtmpx file */
212 /* #undef CONF_WTMPX_FILE */

214 /* Define if you want external askpass support */
215 /* #undef USE_EXTERNAL_ASKPASS */

217 /* Define if libc defines __progname */
218 #define HAVE___PROGNAME 1

220 /* Define if compiler implements __FUNCTION__ */
221 #define HAVE___FUNCTION__ 1

223 /* Define if compiler implements __func__ */
224 #define HAVE___func__ 1

226 /* Define if you want GSS-API support */
227 #define GSSAPI 1

229 /* Define if you have <gssapi/gssapi.h> */
230 #define SUNW_GSSAPI 1

232 /* Define if you have GSS_Store_cred() */
233 #define HAVE_GSS_STORE_CRED  1

235 /* Define if you have __gss_userok() */
236 #define HAVE___GSS_USEROK 1

238 /* Define for simple authorization of GSS-API principals */
239 /* #undef GSSAPI_SIMPLE_USEROK */

241 /* Define if you have gsscred_name_to_unix_cred() (Solaris) */
242 #define HAVE_GSSCRED_API 1

244 /* Define if you have __gss_oid_to_mech() */
245 #define HAVE_GSS_OID_TO_MECH 1

247 /* Define if you have gss_oid_to_str() */
248 #define HAVE_GSS_OID_TO_STR 1

250 /* Define if you want support for MIT krb5 GSS internals */
251 /* #undef KRB5_GSS */

253 /* Define if you want support for GSI GSS internals */
254 /* #undef GSI_GSS */

256 /* Define if you want raw Kerberos 5 support */
257 /* #undef KRB5 */
```

```
259 /* Define if you want GSI/Globus authentication support */
260 /* #undef GSI */

262 /* Define this if you are using the Heimdal version of Kerberos V5 */
263 /* #undef HEIMDAL */

265 /* Define if you want Kerberos 4 support */
266 /* #undef KRB4 */

268 /* Define if you want AFS support */
269 /* #undef AFS */

271 /* Define if you want S/Key support */
272 /* #undef SKEY */

274 /* Define if you want TCP Wrappers support */
275 #define LIBWRAP 1

277 /* Define if your libraries define login() */
278 /* #undef HAVE_LOGIN */

280 /* Define if your libraries define getpagesize() */
281 #define HAVE_GETPAGESIZE 1

283 /* Define if xauth is found in your path */
284 #define XAUTH_PATH "/usr/X11/bin/xauth"

286 /* Define if rsh is found in your path */
287 #define RSH_PATH "/usr/bin/rsh"

289 /* Define if you want to allow MD5 passwords */
290 /* #undef HAVE_MD5_PASSWORDS */

292 /* Define if you want to disable shadow passwords */
293 /* #undef DISABLE_SHADOW */

295 /* Define if you want to use shadow password expire field */
296 /* #undef HAS_SHADOW_EXPIRE */

298 /* Define if you have Digital Unix Security Integration Architecture */
299 /* #undef HAVE_OSF_SIA */

301 /* Define if you have getpwanam(3) [SunOS 4.x] */
302 /* #undef HAVE_GETPWANAM */

304 /* Define if you have an old version of PAM which takes only one argument */
305 /* to pam_strerror */
306 /* #undef HAVE_OLD_PAM */

308 /* Define if you are using Solaris-derived PAM which passes pam_messages  */
309 /* to the conversation function with an extra level of indirection */
310 #define PAM_SUN_CODEBASE 1

312 /* Set this to your mail directory if you don't have maillock.h */
313 /* #undef MAIL_DIRECTORY */

315 /* Data types */
316 #define HAVE_U_INT 1
317 #define HAVE_INTXX_T 1
318 /* #undef HAVE_U_INTXX_T */
319 #define HAVE_UINTXX_T 1
320 #define HAVE_INT64_T 1
321 /* #undef HAVE_U_INT64_T */
322 #define HAVE_U_CHAR 1
323 #define HAVE_SIZE_T 1
324 #define HAVE_SSIZE_T 1
```



```
325 #define HAVE_CLOCK_T 1
326 #define HAVE_MODE_T 1
327 #define HAVE_PID_T 1
328 #define HAVE_SA_FAMILY_T 1
329 #define HAVE_STRUCT_SOCKADDR_STORAGE 1
330 #define HAVE_STRUCT_ADDRINFO 1
331 #define HAVE_STRUCT_IN6_ADDR 1
332 #define HAVE_STRUCT_SOCKADDR_IN6 1

334 /* Fields in struct sockaddr_storage */
335 #define HAVE_SS_FAMILY_IN_SS 1
336 /* #undef HAVE___SS_FAMILY_IN_SS */

338 /* Define if you have /dev/ptmx */
339 #define HAVE_DEV_PTMX 1

341 /* Define if you have /dev/ptc */
342 /* #undef HAVE_DEV_PTS_AND_PTC */

344 /* Define if you need to use IP address instead of hostname in $DISPLAY */
345 /* #undef IPADDR_IN_DISPLAY */

347 /*
348  * Specify the default $PATH. While /bin is a symbolic link to /usr/bin in
349  * Solaris, to include both of them there may help when users use
350  * ChrootDirectory options with plain SSH connections, without their own shell
351  * profiles.
352  */
353 #define USER_PATH "/usr/bin:/bin"

355 /* Specify location of ssh.pid */
356 #define _PATH_SSH_PIDDIR "/var/run"

358 /* Use IPv4 for connection by default, IPv6 can still if explicity asked */
359 /* #undef IPV4_DEFAULT */

361 /* getaddrinfo is broken (if present) */
362 /* #undef BROKEN_GETADDRINFO */

364 /* Workaround more Linux IPv6 quirks */
365 /* #undef DONT_TRY_OTHER_AF */

367 /* Detect IPv4 in IPv6 mapped addresses and treat as IPv4 */
368 #define IPV4_IN_IPV6 1

370 /* Define if you have BSD auth support */
371 /* #undef BSD_AUTH */

373 /* Define if X11 doesn't support AF_UNIX sockets on that system */
374 /* #undef NO_X11_UNIX_SOCKETS */

376 /* Define if the concept of ports only accessible to superusers isn't known */
377 /* #undef NO_IPPORT_RESERVED_CONCEPT */

379 /* Needed for SCO and NeXT */
380 /* #undef BROKEN_SAVED_UIDS */

382 /* Define if your system glob() function has the GLOB_ALTDIRFUNC extension */
383 /* #undef GLOB_HAS_ALTDIRFUNC */
384 #define GLOB_HAS_ALTDIRFUNC     1

386 /* Define if your system glob() function has gl_matchc options in glob_t */
387 /* #undef GLOB_HAS_GL_MATCHC */
388 #define GLOB_HAS_GL_MATCHC      1

390 /*
```

```
391  * Define in your struct dirent expects you to allocate extra space for
392  * d_name
393  */
394 #define BROKEN_ONE_BYTE_DIRENT_D_NAME 1

396 /* Define if your getopt(3) defines and uses optreset */
397 /* #undef HAVE_GETOPT_OPTRESET */

399 /* Define on *nto-qnx systems */
400 /* #undef MISSING_NFDBITS */

402 /* Define on *nto-qnx systems */
403 /* #undef MISSING_HOWMANY */

405 /* Define on *nto-qnx systems */
406 /* #undef MISSING_FD_MASK */

408 /*
409  * Use libedit or libtecla for sftp
410  * If both USE_LIBEDIT and USE_LIBTECLA are defined, then USE_LIBEDIT will
411  * have higher precedence.
412  */
413 #undef USE_LIBEDIT
414 #define USE_LIBTECLA    1

416 /* Define if you want to use OpenSSL's internally seeded PRNG only */
417 #define OPENSSL_PRNG_ONLY 1

419 /* Define if you shouldn't strip 'tty' from your ttyname in [uw]tmp */
420 /* #undef WITH_ABBREV_NO_TTY */

422 /* Define if you want a different $PATH for the superuser */
423 #define SUPERUSER_PATH "/usr/sbin:/usr/bin"

425 /* Path that unprivileged child will chroot() to in privep mode */
426 /* #undef PRIVSEP_PATH */

428 /* Define if your platform needs to skip post auth file descriptor passing */
429 /* #undef DISABLE_FD_PASSING */


432 /* Define to 1 if the 'getpgrp' function requires zero arguments. */
433 #define GETPGRP_VOID 1

435 /* Define to 1 if you have the 'arc4random' function. */
436 /* #undef HAVE_ARC4RANDOM */

438 /* Define to 1 if you have the 'asprintf' function. */
439 #define HAVE_ASPRINTF 1

441 /* Define to 1 if you have the 'b64_ntop' function. */
442 /* #undef HAVE_B64_NTOP */

444 /* Define to 1 if you have the 'bcopy' function. */
445 #define HAVE_BCOPY 1

447 /* Define to 1 if you have the 'bindresvport_sa' function. */
448 /* #undef HAVE_BINDRESVPORT_SA */

450 /* Define to 1 if you have the <bstring.h> header file. */
451 /* #undef HAVE_BSTRING_H */

453 /* Define to 1 if you have the 'clock' function. */
454 #define HAVE_CLOCK 1

456 /* Define to 1 if you have the <crypt.h> header file. */
```



```
457 #define HAVE_CRYPT_H 1

459 /* Define to 1 if you have the 'dirname' function. */
460 #define HAVE_DIRNAME 1

462 /* Define to 1 if you have the <endian.h> header file. */
463 /* #undef HAVE_ENDIAN_H */

465 /* Define to 1 if you have the 'endutent' function. */
466 #define HAVE_ENDUTENT 1

468 /* Define to 1 if you have the 'endutxent' function. */
469 #define HAVE_ENDUTXENT 1

471 /* Define to 1 if you have the 'fchmod' function. */
472 #define HAVE_FCHMOD 1

474 /* Define to 1 if you have the 'fchown' function. */
475 #define HAVE_FCHOWN 1

477 /* Define to 1 if you have the <floatingpoint.h> header file. */
478 #define HAVE_FLOATINGPOINT_H 1

480 /* Define to 1 if you have the 'freeaddrinfo' function. */
481 #define HAVE_FREEADDRINFO 1

483 /* Define to 1 if you have the 'futimes' function. */
484 /* #undef HAVE_FUTIMES */

486 /* Define to 1 if you have the 'gai_strerror' function. */
487 #define HAVE_GAI_STRERROR 1

489 /* Define to 1 if you have the 'getaddrinfo' function. */
490 #define HAVE_GETADDRINFO 1

492 /* Define to 1 if you have the 'getcwd' function. */
493 #define HAVE_GETCWD 1

495 /* Define to 1 if you have the 'getgrouplist' function. */
496 /* #undef HAVE_GETGROUPLIST */

498 /* Define to 1 if you have the 'getluid' function. */
499 /* #undef HAVE_GETLUID */

501 /* Define to 1 if you have the 'getnameinfo' function. */
502 #define HAVE_GETNAMEINFO 1

504 /* Define to 1 if you have the 'getopt' function. */
505 #define HAVE_GETOPT 1

507 /* Define to 1 if you have the <getopt.h> header file. */
508 /* #undef HAVE_GETOPT_H */

510 /* Define to 1 if you have the 'getpeereid' function. */
511 /* #undef HAVE_GETPEEREID */

513 /* Define to 1 if you have the 'getpeerucred' function. */
514 #define HAVE_GETPEERUCRED 1

516 /* Define to 1 if you have the 'getpwanam' function. */
517 /* #undef HAVE_GETPWANAM */

519 /* Define to 1 if you have the 'getrlimit' function. */
520 #define HAVE_GETRLIMIT 1

522 /* Define to 1 if you have the 'getrusage' function. */
```

```
523 #define HAVE_GETRUSAGE 1

525 /* Define to 1 if you have the 'gettimeofday' function. */
526 #define HAVE_GETTIMEOFDAY 1

528 /* Define to 1 if you have the 'getttyent' function. */
529 /* #undef HAVE_GETTTYENT */

531 /* Define to 1 if you have the 'getutent' function. */
532 #define HAVE_GETUTENT 1

534 /* Define to 1 if you have the 'getutid' function. */
535 #define HAVE_GETUTID 1

537 /* Define to 1 if you have the 'getutline' function. */
538 #define HAVE_GETUTLINE 1

540 /* Define to 1 if you have the 'getutxent' function. */
541 #define HAVE_GETUTXENT 1

543 /* Define to 1 if you have the 'getutxid' function. */
544 #define HAVE_GETUTXID 1

546 /* Define to 1 if you have the 'getutxline' function. */
547 #define HAVE_GETUTXLINE 1

549 /* Define to 1 if you have the 'glob' function. */
550 #define HAVE_GLOB 1

552 /* Define to 1 if you have the <glob.h> header file. */
553 #define HAVE_GLOB_H 1

555 /* Define to 1 if you have the <ia.h> header file. */
556 /* #undef HAVE_IA_H */

558 /* Define to 1 if you have the 'inet_aton' function. */
559 /* #undef HAVE_INET_ATON */

561 /* Define to 1 if you have the 'inet_ntoa' function. */
562 #define HAVE_INET_NTOA 1

564 /* Define to 1 if you have the 'inet_ntop' function. */
565 #define HAVE_INET_NTOP 1

567 /* Define to 1 if you have the 'innetgr' function. */
568 #define HAVE_INNETGR 1

570 /* Define to 1 if you have the <inttypes.h> header file. */
571 #define HAVE_INTTYPES_H 1

573 /* Define to 1 if you have the <krb.h> header file. */
574 /* #undef HAVE_KRB_H */

576 /* Define to 1 if you have the <lastlog.h> header file. */
577 #define HAVE_LASTLOG_H 1

579 /* Define to 1 if you have the 'crypt' library (-lcrypt). */
580 /* #undef HAVE_LIBCRYPT */

582 /* Define to 1 if you have the 'des' library (-ldes). */
583 /* #undef HAVE_LIBDES */

585 /* Define to 1 if you have the 'des425' library (-ldes425). */
586 /* #undef HAVE_LIBDES425 */

588 /* Define to 1 if you have the 'dl' library (-ldl). */
```



```
589 #define HAVE_LIBDL 1

591 /* Define to 1 if you have the <libgen.h> header file. */
592 #define HAVE_LIBGEN_H 1

594 /* Define to 1 if you have the 'krb' library (-lkrb). */
595 /* #undef HAVE_LIBKRB */

597 /* Define to 1 if you have the 'krb4' library (-lkrb4). */
598 /* #undef HAVE_LIBKRB4 */

600 /* Define to 1 if you have the 'nsl' library (-lnsl). */
601 #define HAVE_LIBNSL 1

603 /* Define to 1 if you have the 'pam' library (-lpam). */
604 #define HAVE_LIBPAM 1

606 /* Define to 1 if you have the 'resolv' library (-lresolv). */
607 /* #undef HAVE_LIBRESOLV */

609 /* Define to 1 if you have the 'sectok' library (-lsectok). */
610 /* #undef HAVE_LIBSECTOK */

612 /* Define to 1 if you have the 'socket' library (-lsocket). */
613 #define HAVE_LIBSOCKET 1

615 /* Define to 1 if you have the <libutil.h> header file. */
616 /* #undef HAVE_LIBUTIL_H */

618 /* Define to 1 if you have the 'xnet' library (-lxnet). */
619 /* #undef HAVE_LIBXNET */

621 /* Define to 1 if you have the 'z' library (-lz). */
622 #define HAVE_LIBZ 1

624 /* Define to 1 if you have the <limits.h> header file. */
625 #define HAVE_LIMITS_H 1

627 /* Define to 1 if you have the <login.h> header file. */
628 /* #undef HAVE_LOGIN_H */

630 /* Define to 1 if you have the 'logout' function. */
631 /* #undef HAVE_LOGOUT */

633 /* Define to 1 if you have the 'logwtmp' function. */
634 /* #undef HAVE_LOGWTMP */

636 /* Define to 1 if you have the <maillock.h> header file. */
637 #define HAVE_MAILLOCK_H 1

639 /* Define to 1 if you have the 'md5_crypt' function. */
640 /* #undef HAVE_MD5_CRYPT */

642 /* Define to 1 if you have the 'memmove' function. */
643 #define HAVE_MEMMOVE 1

645 /* Define to 1 if you have the <memory.h> header file. */
646 #define HAVE_MEMORY_H 1

648 /* Define to 1 if you have mkstemp, mkstemps and mkdtemp */
649 #define HAVE_MKDTEMP 1

651 /* Define to 1 if you have the 'mmap' function. */
652 #define HAVE_MMAP 1

654 /* Define to 1 if you have the <netdb.h> header file. */
```

```
655 #define HAVE_NETDB_H 1

657 /* Define to 1 if you have the <netgroup.h> header file. */
658 /* #undef HAVE_NETGROUP_H */

660 /* Define to 1 if you have the <netinet/in_systm.h> header file. */
661 #define HAVE_NETINET_IN_SYSTM_H 1

663 /* Define to 1 if you have the 'ngetaddrinfo' function. */
664 /* #undef HAVE_NGETADDRINFO */

666 /* Define to 1 if you have the 'ogetaddrinfo' function. */
667 /* #undef HAVE_OGETADDRINFO */

669 /* Define to 1 if you have the 'openpty' function. */
670 /* #undef HAVE_OPENPTY */

672 /* Define to 1 if you have the 'pam_getenvlist' function. */
673 #define HAVE_PAM_GETENVLIST 1

675 /* Define to 1 if you have the <paths.h> header file. */
676 /* #undef HAVE_PATHS_H */

678 /* Define to 1 if you have the <pty.h> header file. */
679 /* #undef HAVE_PTY_H */

681 /* Define to 1 if you have the 'pututline' function. */
682 #define HAVE_PUTUTLINE 1

684 /* Define to 1 if you have the 'pututxline' function. */
685 #define HAVE_PUTUTXLINE 1

687 /* Define to 1 if you have the 'readpassphrase' function. */
688 /* #undef HAVE_READPASSPHRASE */

690 /* Define to 1 if you have the <readpassphrase.h> header file. */
691 /* #undef HAVE_READPASSPHRASE_H */

693 /* Define to 1 if you have the 'realpath' function. */
694 #define HAVE_REALPATH 1

696 /* Define to 1 if you have the 'recvmsg' function. */
697 #define HAVE_RECVMSG 1

699 /* Define to 1 if you have the <rpc/types.h> header file. */
700 #define HAVE_RPC_TYPES_H 1

702 /* Define to 1 if you have the 'rresvport_af' function. */
703 #define HAVE_RRESVPORT_AF 1

705 /* Define to 1 if you have the <sectok.h> header file. */
706 /* #undef HAVE_SECTOK_H */

708 /* Define to 1 if you have the <security/pam_appl.h> header file. */
709 #define HAVE_SECURITY_PAM_APPL_H 1

711 /* Define to 1 if you have the 'sendmsg' function. */
712 #define HAVE_SENDMSG 1

714 /* Define to 1 if you have the 'setdtablesize' function. */
715 /* #undef HAVE_SETDTABLESIZE */

717 /* Define to 1 if you have the 'setegid' function. */
718 #define HAVE_SETEGID 1

720 /* Define to 1 if you have the 'setenv' function. */
```



```
721 #define HAVE_SETENV 1

723 /* Define to 1 if you have the 'seteuid' function. */
724 #define HAVE_SETEUID 1

726 /* Define to 1 if you have the 'setgroups' function. */
727 #define HAVE_SETGROUPS 1

729 /* Define to 1 if you have the 'setlogin' function. */
730 /* #undef HAVE_SETLOGIN */

732 /* Define to 1 if you have the 'setluid' function. */
733 /* #undef HAVE_SETLUID */

735 /* Define to 1 if you have the 'setpcred' function. */
736 /* #undef HAVE_SETPCRED */

738 /* Define to 1 if you have the 'setproctitle' function. */
739 /* #undef HAVE_SETPROCTITLE */

741 /* Define to 1 if you have the 'setresgid' function. */
742 /* #undef HAVE_SETRESGID */

744 /* Define to 1 if you have the 'setreuid' function. */
745 #define HAVE_SETREUID 1

747 /* Define to 1 if you have the 'setrlimit' function. */
748 #define HAVE_SETRLIMIT 1

750 /* Define to 1 if you have the 'setsid' function. */
751 #define HAVE_SETSID 1

753 /* Define to 1 if you have the 'setutent' function. */
754 #define HAVE_SETUTENT 1

756 /* Define to 1 if you have the 'setutxent' function. */
757 #define HAVE_SETUTXENT 1

759 /* Define to 1 if you have the 'setvbuf' function. */
760 #define HAVE_SETVBUF 1

762 /* Define to 1 if you have the <shadow.h> header file. */
763 #define HAVE_SHADOW_H 1

765 /* Define to 1 if you have the 'sigaction' function. */
766 #define HAVE_SIGACTION 1

768 /* Define to 1 if you have the 'sigvec' function. */
769 /* #undef HAVE_SIGVEC */

771 /* Define to 1 if the system has the type 'sig_atomic_t'. */
772 #define HAVE_SIG_ATOMIC_T 1

774 /* Define to 1 if you have the 'snprintf' function. */
775 #define HAVE_SNPRINTF 1

777 /* Define to 1 if you have the 'socketpair' function. */
778 #define HAVE_SOCKETPAIR 1

780 /* Define to 1 if you have the <stddef.h> header file. */
781 #define HAVE_STDDEF_H 1

783 /* Define to 1 if you have the <stdint.h> header file. */
784 /* #undef HAVE_STDINT_H */

786 /* Define to 1 if you have the <stdlib.h> header file. */
```

```
787 #define HAVE_STDLIB_H 1

789 /* Define to 1 if you have the 'strerror' function. */
790 #define HAVE_STRERROR 1

792 /* Define to 1 if you have the 'strftime' function. */
793 #define HAVE_STRFTIME 1

795 /* Define to 1 if you have the <strings.h> header file. */
796 #define HAVE_STRINGS_H 1

798 /* Define to 1 if you have the <string.h> header file. */
799 #define HAVE_STRING_H 1

801 /* Define to 1 if you have the 'strlcat' function. */
802 #define HAVE_STRLCAT 1

804 /* Define to 1 if you have the 'strlcpy' function. */
805 #define HAVE_STRLCPY 1

807 /* Define to 1 if you have the 'strmode' function. */
808 /* #undef HAVE_STRMODE */

810 /* Define to 1 if 'st_blksize' is member of 'struct stat'. */
811 #define HAVE_STRUCT_STAT_ST_BLKSIZE 1

813 /* Define to 1 if you have the 'sysconf' function. */
814 #define HAVE_SYSCONF 1

816 /* Define to 1 if you have the <sys/bitypes.h> header file. */
817 /* #undef HAVE_SYS_BITYPES_H */

819 /* Define to 1 if you have the <sys/bsdtty.h> header file. */
820 /* #undef HAVE_SYS_BSDTTY_H */

822 /* Define to 1 if you have the <sys/cdefs.h> header file. */
823 /* #undef HAVE_SYS_CDEFS_H */


826 /* Define to 1 if you have the <sys/mman.h> header file. */
827 #define HAVE_SYS_MMAN_H 1

829 /* Define to 1 if you have the <sys/select.h> header file. */
830 #define HAVE_SYS_SELECT_H 1

832 /* Define to 1 if you have the <sys/stat.h> header file. */
833 #define HAVE_SYS_STAT_H 1

835 /* Define to 1 if you have the <sys/stropts.h> header file. */
836 #define HAVE_SYS_STROPTS_H 1

838 /* Define to 1 if you have the <sys/sysmacros.h> header file. */
839 #define HAVE_SYS_SYSMACROS_H 1

841 /* Define to 1 if you have the <sys/time.h> header file. */
842 #define HAVE_SYS_TIME_H 1

844 /* Define to 1 if you have the <sys/types.h> header file. */
845 #define HAVE_SYS_TYPES_H 1

847 /* Define to 1 if you have the <sys/un.h> header file. */
848 #define HAVE_SYS_UN_H 1

850 /* Define to 1 if you have the 'tcgetpgrp' function. */
851 #define HAVE_TCGETPGRP 1
```



```
853 /* Define to 1 if you have the 'time' function. */
854 #define HAVE_TIME 1

856 /* Define to 1 if you have the <time.h> header file. */
857 #define HAVE_TIME_H 1

859 /* Define to 1 if you have the <tmpdir.h> header file. */
860 /* #undef HAVE_TMPDIR_H */

862 /* Define to 1 if you have the 'truncate' function. */
863 #define HAVE_TRUNCATE 1

865 /* Define to 1 if you have the <ttyent.h> header file. */
866 /* #undef HAVE_TTYENT_H */

868 /* Define to 1 if you have the <ucred.h> header file. */
869 #define HAVE_UCRED_H 1

871 /* Define to 1 if you have the <unistd.h> header file. */
872 #define HAVE_UNISTD_H 1

874 /* Define to 1 if you have the 'updwtmp' function. */
875 #define HAVE_UPDWTMP 1

877 /* Define to 1 if you have the <usersec.h> header file. */
878 /* #undef HAVE_USERSEC_H */

880 /* Define to 1 if you have the <util.h> header file. */
881 /* #undef HAVE_UTIL_H */

883 /* Define to 1 if you have the 'utimes' function. */
884 #define HAVE_UTIMES 1

886 /* Define to 1 if you have the <utime.h> header file. */
887 #define HAVE_UTIME_H 1

889 /* Define to 1 if you have the 'utmpname' function. */
890 #define HAVE_UTMPNAME 1

892 /* Define to 1 if you have the 'utmpxname' function. */
893 #define HAVE_UTMPXNAME 1

895 /* Define to 1 if you have the <utmpx.h> header file. */
896 #define HAVE_UTMPX_H 1

898 /* Define to 1 if you have the <utmp.h> header file. */
899 #define HAVE_UTMP_H 1

901 /* Define to 1 if you have the 'vasprintf' function. */
902 #define HAVE_VASPRINTF 1

904 /* Define to 1 if you have the 'vhangup' function. */
905 #define HAVE_VHANGUP 1

907 /* Define to 1 if you have the 'vsnprintf' function. */
908 #define HAVE_VSNPRINTF 1

910 /* Define to 1 if you have the 'waitpid' function. */
911 #define HAVE_WAITPID 1

913 /* Define to 1 if you have the '_getpty' function. */
914 /* #undef HAVE__GETPTY */

916 /* Define to 1 if you have the '__b64_ntop' function. */
917 /* #undef HAVE___B64_NTOP */
```

```
 919 /* Define to the address where bug reports for this package should be sent. */
 920 #define PACKAGE_BUGREPORT ""

 922 /* Define to the full name of this package. */
 923 #define PACKAGE_NAME ""

 925 /* Define to the full name and version of this package. */
 926 #define PACKAGE_STRING ""

 928 /* Define to the one symbol short name of this package. */
 929 #define PACKAGE_TARNAME ""

 931 /* Define to the version of this package. */
 932 #define PACKAGE_VERSION ""

 934 /* The size of a 'char', as computed by sizeof. */
 935 #define SIZEOF_CHAR 1

 937 /* The size of a 'int', as computed by sizeof. */
 938 #define SIZEOF_INT 4

 940 /* The size of a 'long int', as computed by sizeof. */
 941 #define SIZEOF_LONG_INT 4

 943 /* The size of a 'long long int', as computed by sizeof. */
 944 #define SIZEOF_LONG_LONG_INT 8

 946 /* The size of a 'short int', as computed by sizeof. */
 947 #define SIZEOF_SHORT_INT 2

 949 /* Define to 1 if you have the ANSI C header files. */
 950 #define STDC_HEADERS 1

 952 /*
 953  * Define to 1 if your processor stores words with the most significant byte
 954  * first (like Motorola and SPARC, unlike Intel and VAX).
 955  */
 956 #define WORDS_BIGENDIAN 1

 958 /* Number of bits in a file offset, on hosts where this is settable. */
 959 #define _FILE_OFFSET_BITS 64

 961 /* Define for large files, on AIX-style hosts. */
 962 /* #undef _LARGE_FILES */

 964 /*
 965  * Define as '__inline' if that's what the C compiler calls it, or to nothing if
 966  * it is not supported.
 967  */
 968 /* #undef inline */

 970 /* type to use in place of socklen_t if not defined */
 971 /* #undef socklen_t */

 973 /* Define for BSM auditing (Solaris) support */
 974 #define HAVE_BSM 1

 976 /* Define if compiling in ON */
 977 #define SUNW_SSH 1

 979 /* ****************** Shouldn't need to edit below this line ************** */

 981 #ifdef __cplusplus
 982 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    5560 Mon Jan 28 14:19:14 2013
new/usr/src/head/glob.h
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
/*
 * 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 (c) 1989, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Guido van Rossum.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *      @(#)glob.h      8.1 (Berkeley) 6/2/93
 */

/*
 * Copyright 2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright (c) 2012 Gary Mills
 */
```



```
 */

/*
 * Copyright 1985, 1992 by Mortice Kern Systems Inc.  All rights reserved.
 */

#ifndef _GLOB_H
#define _GLOB_H

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

#include <sys/feature_tests.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>

#ifdef  __cplusplus
extern "C" {
#endif

struct stat;

typedef struct  glob_t  {
        /* Members required by POSIX */
        size_t  gl_pathc;       /* Total count of paths matched by pattern */
        size_t  gl_pathc;               /* Count of paths matched by pattern */
        char    **gl_pathv;     /* List of matched pathnames */
        size_t  gl_offs;        /* # of slots reserved in gl_pathv */
        /* Non-POSIX extensions, from Openbsd */
        int gl_matchc;          /* Count of paths matching pattern. */
        int gl_flags;           /* Copy of flags parameter to glob. */
        /* Members only accessed when Non-POSIX flags are specified. */
        struct stat **gl_statv; /* Stat entries corresponding to gl_pathv */
        /*
         * Alternate filesystem access methods for glob; replacement
         * versions of closedir(3), readdir(3), opendir(3), stat(2)
         * and lstat(2).
         */
        void (*gl_closedir)(void *);
        struct dirent *(*gl_readdir)(void *);
        void *(*gl_opendir)(const char *);
        int (*gl_lstat)(const char *, struct stat *);
        int (*gl_stat)(const char *, struct stat *);
        /* following are internal to the implementation */
        char    **gl_pathp;             /* gl_pathv + gl_offs */
        int     gl_pathn;               /* # of elements allocated */
}       glob_t;

/*
 * POSIX "flags" argument to glob function.
 * "flags" argument to glob function.
 */
#define GLOB_ERR        0x0001          /* Don't continue on directory error */
#define GLOB_MARK       0x0002          /* Mark directories with trailing / */
#define GLOB_NOSORT     0x0004          /* Don't sort pathnames */
#define GLOB_NOCHECK    0x0008          /* Return unquoted arg if no match */
#define GLOB_DOOFFS     0x0010          /* Ignore gl_offs unless set */
#define GLOB_APPEND     0x0020          /* Append to previous glob_t */
#define GLOB_NOESCAPE   0x0040          /* Backslashes do not quote M-chars */

/*
 * Non-POSIX "flags" argument to glob function, from Openbsd.
 */
#define GLOB_BRACE      0x0080  /* Expand braces ala csh. */
#define GLOB_MAGCHAR    0x0100  /* Pattern had globbing characters. */
#define GLOB_NOMAGIC    0x0200  /* GLOB_NOCHECK without magic chars (csh). */
```

```
120 #define GLOB_QUOTE      0x0400  /* Quote special chars with \. */
121 #define GLOB_TILDE      0x0800  /* Expand tilde names from the passwd file. */
122 #define GLOB_LIMIT      0x2000  /* Limit pattern match output to ARG_MAX */
123 #define GLOB_KEEPSTAT   0x4000  /* Retain stat data for paths in gl_statv. */
124 #define GLOB_ALTDIRFUNC 0x8000  /* Use alternately specified directory funcs. */

126 /*
127  * Error returns from "glob"
128  */
129 #define GLOB_NOSYS      (-4)            /* function not supported (XPG4) */
130 #define GLOB_NOMATCH    (-3)            /* Pattern does not match */
131 #define GLOB_NOSPACE    (-2)            /* Not enough memory */
132 #define GLOB_ABORTED    (-1)            /* GLOB_ERR set or errfunc return!=0 */
133 #define GLOB_ABEND      GLOB_ABORTED    /* backward compatibility */

135 #if defined(__STDC__)
136 extern int glob(const char *_RESTRICT_KYWD, int, int(*)(const char *, int),
137                 glob_t *_RESTRICT_KYWD);
138 extern void globfree(glob_t *);
139 #else
140 extern int glob();
141 extern void globfree();
142 #endif

144 #ifdef  __cplusplus
145 }
```

*unchanged_portion_omitted_*

****
1774 Mon Jan 28 14:19:15 2013
new/usr/src/lib/libc/port/regex/THIRDPARTYLICENSE
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
****

```
The following files from the C library:
    ./glob.c
are provided under the following terms:

Copyright (c) 2013 Gary Mills
Copyright (c) 1989, 1993
        The Regents of the University of California.  All rights reserved.

$OpenBSD: glob.c,v 1.39 2012/01/20 07:09:42 tedu Exp $

This code is derived from software contributed to Berkeley by
Guido van Rossum.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. Neither the name of the University nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```

```
**********************************************************
      25 Mon Jan 28 14:19:15 2013
new/usr/src/lib/libc/port/regex/THIRDPARTYLICENSE.descrip
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
```

ALL OF THE GLOB FUNCTION

```
**********************************************************
   30122 Mon Jan 28 14:19:16 2013
new/usr/src/lib/libc/port/regex/glob.c
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
/*
 * Copyright (c) 2012 Gary Mills
 */
/*      $OpenBSD: glob.c,v 1.39 2012/01/20 07:09:42 tedu Exp $ */
/*
 * Copyright (c) 1989, 1993
 *      The Regents of the University of California.  All rights reserved.
 * CDDL HEADER START
 *
 * This code is derived from software contributed to Berkeley by
 * Guido van Rossum.
 * 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.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 * 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.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 * 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
 */

/*
 * glob(3) -- a superset of the one defined in POSIX 1003.2.
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * This code is MKS code ported to Solaris originally with minimum
 * modifications so that upgrades from MKS would readily integrate.
```



```
 * The MKS basis for this modification was:
 *
 * The [!...] convention to negate a range is supported (SysV, Posix, ksh).
 *      $Id: glob.c 1.31 1994/04/07 22:50:43 mark
 *
 * Optional extra services, controlled by flags not defined by POSIX:
 * Additional modifications have been made to this code to make it
 * 64-bit clean.
 */

/*
 * glob, globfree -- POSIX.2 compatible file name expansion routines.
 *
 * GLOB_QUOTE:
 *      Escaping convention: \ inhibits any special meaning the following
 *      character might have (except \ at end of string is retained).
 * GLOB_MAGCHAR:
 *      Set in gl_flags if pattern contained a globbing character.
 * GLOB_NOMAGIC:
 *      Same as GLOB_NOCHECK, but it will only append pattern if it did
 *      not contain any magic characters.  [Used in csh style globbing]
 * GLOB_ALTDIRFUNC:
 *      Use alternately specified directory access functions.
 * GLOB_TILDE:
 *      expand ~user/foo to the /home/dir/of/user/foo
 * GLOB_BRACE:
 *      expand {1,2}{a,b} to 1a 1b 2a 2b
 * gl_matchc:
 *      Number of matches in the current invocation of glob.
 * Copyright 1985, 1991 by Mortice Kern Systems Inc.  All rights reserved.
 *
 * Written by Eric Gisin.
 */

#include <sys/param.h>
#include <sys/stat.h>
#pragma ident   "%Z%%M% %I%     %E% SMI"

#include <ctype.h>
#include <dirent.h>
#include <errno.h>
#include <glob.h>
#include <limits.h>
#include <pwd.h>
#pragma weak _glob = glob
#pragma weak _globfree = globfree

#include "lint.h"
#include <stdio.h>
#include <unistd.h>
#include <limits.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <wchar.h>
#include <wctype.h>
#include <dirent.h>
#include <sys/stat.h>
#include <glob.h>
#include <errno.h>
#include <fnmatch.h>

#define DOLLAR          '$'
#define DOT             '.'
#define EOS             '\0'
#define LBRACKET        '['
```

```
 82 #define NOT             '!'
 83 #define QUESTION        '?'
 84 #define QUOTE           '\\'
 85 #define RANGE           '-'
 86 #define RBRACKET        ']'
 87 #define SEP             '/'
 88 #define STAR            '*'
 89 #define TILDE           '~'
 90 #define UNDERSCORE      '_'
 91 #define LBRACE          '{'
 92 #define RBRACE          '}'
 93 #define SLASH           '/'
 94 #define COMMA           ','
 95 #define COLON           ':'
 63 #define GLOB__CHECK     0x80    /* stat generated paths */

 97 #define M_QUOTE         0x800000
 98 #define M_PROTECT       0x400000
 65 #define INITIAL 8               /* initial pathv allocation */
 66 #define NULLCPP ((char **)0)    /* Null char ** */
 67 #define NAME_MAX        1024    /* something large */

100 typedef struct wcat {
101         wchar_t w_wc;
102         uint_t w_at;
103 } wcat_t;
 69 static int      globit(size_t, const char *, glob_t *, int,
 70         int (*)(const char *, int), char **);
 71 static int      pstrcmp(const void *, const void *);
 72 static int      append(glob_t *, const char *);

105 #define M_ALL           '*'     /* Plus M_QUOTE */
106 #define M_END           ']'     /* Plus M_QUOTE */
107 #define M_NOT           '!'     /* Plus M_QUOTE */
108 #define M_ONE           '?'     /* Plus M_QUOTE */
109 #define M_RNG           '-'     /* Plus M_QUOTE */
110 #define M_SET           '['     /* Plus M_QUOTE */
111 #define M_CLASS         ':'     /* Plus M_QUOTE */
112 #define ismeta(c)       (((c).w_at&M_QUOTE) != 0)

114 #define GLOB_LIMIT_MALLOC       65536
115 #define GLOB_LIMIT_STAT         2048
116 #define GLOB_LIMIT_READDIR      16384

118 /* Limit of recursion during matching attempts. */
119 #define GLOB_LIMIT_RECUR        64

121 struct glob_lim {
122         size_t  glim_malloc;
123         size_t  glim_stat;
124         size_t  glim_readdir;
125 };

127 struct glob_path_stat {
128         char            *gps_path;
129         struct stat     *gps_stat;
130 };

132 static int       compare(const void *, const void *);
133 static int       compare_gps(const void *, const void *);
134 static int       g_Ctoc(const wcat_t *, char *, uint_t);
135 static int       g_lstat(wcat_t *, struct stat *, glob_t *);
136 static DIR      *g_opendir(wcat_t *, glob_t *);
137 static wcat_t   *g_strchr(const wcat_t *, wchar_t);
138 static int       g_stat(wcat_t *, struct stat *, glob_t *);
139 static int       glob0(const wcat_t *, glob_t *, struct glob_lim *,
```



```
140                         int (*)(const char *, int));
141 static int       glob1(wcat_t *, wcat_t *, glob_t *, struct glob_lim *,
142                         int (*)(const char *, int));
143 static int       glob2(wcat_t *, wcat_t *, wcat_t *, wcat_t *, wcat_t *,
144                         wcat_t *, glob_t *, struct glob_lim *,
145                         int (*)(const char *, int));
146 static int       glob3(wcat_t *, wcat_t *, wcat_t *, wcat_t *, wcat_t *,
147                         wcat_t *, wcat_t *, glob_t *, struct glob_lim *,
148                         int (*)(const char *, int));
149 static int       globextend(const wcat_t *, glob_t *, struct glob_lim *,
150                         struct stat *);
151 static
152 const wcat_t    *globtilde(const wcat_t *, wcat_t *, size_t, glob_t *);
153 static int       globexp1(const wcat_t *, glob_t *, struct glob_lim *,
154                         int (*)(const char *, int));
155 static int       globexp2(const wcat_t *, const wcat_t *, glob_t *,
156                         struct glob_lim *, int (*)(const char *, int));
157 static int       match(wcat_t *, wcat_t *, wcat_t *, int);
158 #ifdef DEBUG
159 static void      qprintf(const char *, wcat_t *);
160 #endif

162 int
163 glob(const char *pattern, int flags, int (*errfunc)(const char *, int),
164     glob_t *pglob)
165 {
166         const char *patnext;
167         size_t n;
168         wchar_t c;
169         wcat_t *bufnext, *bufend, patbuf[MAXPATHLEN];
170         struct glob_lim limit = { 0, 0, 0 };

172         if (strnlen(pattern, PATH_MAX) == PATH_MAX)
173                 return (GLOB_NOMATCH);

175         patnext = pattern;
176         if (!(flags & GLOB_APPEND)) {
177                 pglob->gl_pathc = 0;
178                 pglob->gl_pathv = NULL;
179                 if ((flags & GLOB_KEEPSTAT) != 0)
180                         pglob->gl_statv = NULL;
181                 if (!(flags & GLOB_DOOFFS))
182                         pglob->gl_offs = 0;
183         }
184         pglob->gl_flags = flags & ~GLOB_MAGCHAR;
185         pglob->gl_matchc = 0;

187         if (pglob->gl_offs < 0 || pglob->gl_pathc < 0 ||
188             pglob->gl_offs >= INT_MAX || pglob->gl_pathc >= INT_MAX ||
189             pglob->gl_pathc >= INT_MAX - pglob->gl_offs - 1)
190                 return (GLOB_NOSPACE);

192         bufnext = patbuf;
193         bufend = bufnext + MAXPATHLEN - 1;
194         if (flags & GLOB_NOESCAPE) {
195                 while (bufnext < bufend) {
196                         if ((n = mbtowc(&c, patnext, MB_LEN_MAX)) > 0) {
197                                 patnext += n;
198                                 bufnext->w_at = 0;
199                                 (bufnext++)->w_wc = c;
200                         } else if (n == 0) {
201                                 break;
202                         } else {
203                                 return (GLOB_NOMATCH);
204                         }
205                 }
```

```
206         } else {
207                 /* Protect the quoted characters. */
208                 while (bufnext < bufend) {
209                         if ((n = mbtowc(&c, patnext, MB_LEN_MAX)) > 0) {
210                                 patnext += n;
211                                 if (c == QUOTE) {
212                                         n = mbtowc(&c, patnext, MB_LEN_MAX);
213                                         if (n < 0)
214                                                 return (GLOB_NOMATCH);
215                                         if (n > 0)
216                                                 patnext += n;
217                                         if (n == 0)
218                                                 c = QUOTE;
219                                         bufnext->w_at = M_PROTECT;
220                                         (bufnext++)->w_wc = c;
221                                 } else {
222                                         bufnext->w_at = 0;
223                                         (bufnext++)->w_wc = c;
224                                 }
225                         } else if (n == 0) {
226                                 break;
227                         } else {
228                                 return (GLOB_NOMATCH);
229                         }
230                 }
231         }
232         bufnext->w_at = 0;
233         bufnext->w_wc = EOS;
234
235         if (flags & GLOB_BRACE)
236                 return (globexp1(patbuf, pglob, &limit, errfunc));
237         else
238                 return (glob0(patbuf, pglob, &limit, errfunc));
239 }
240
241 /*
242  * Expand recursively a glob {} pattern. When there is no more expansion
243  * invoke the standard globbing routine to glob the rest of the magic
244  * characters
245  */
246 static int
247 globexp1(const wcat_t *pattern, glob_t *pglob, struct glob_lim *limitp,
248     int (*errfunc)(const char *, int))
249 {
250         const wcat_t *ptr = pattern;
251         size_t i;
252
253         /* Protect a single {}, for find(1), like csh */
254         if (pattern[0].w_wc == LBRACE && pattern[1].w_wc == RBRACE &&
255             pattern[2].w_wc == EOS)
256                 return (glob0(pattern, pglob, limitp, errfunc));
257         if (gp->gl_pathv == 0)
258                 return;
259
260         if ((ptr = (const wcat_t *) g_strchr(ptr, LBRACE)) != NULL)
261                 return (globexp2(ptr, pattern, pglob, limitp, errfunc));
262         for (i = gp->gl_offs; i < gp->gl_offs + gp->gl_pathc; ++i)
263                 free(gp->gl_pathv[i]);
264         free((void *)gp->gl_pathv);
265
266         return (glob0(pattern, pglob, limitp, errfunc));
267         gp->gl_pathc = 0;
268         gp->gl_pathv = NULLCPP;
```



```
269 }
270
271 /*
272  * Recursive brace globbing helper. Tries to expand a single brace.
273  * If it succeeds then it invokes globexp1 with the new pattern.
274  * If it fails then it tries to glob the rest of the pattern and returns.
275  * Do filename expansion.
276  */
277 static int
278 globexp2(const wcat_t *ptr, const wcat_t *pattern, glob_t *pglob,
279     struct glob_lim *limitp, int (*errfunc)(const char *, int))
280 int
281 glob(const char *pattern, int flags,
282         int (*errfn)(const char *, int), glob_t *gp)
283 {
284         int     i, rv;
285         wcat_t   *lm, *ls;
286         const wcat_t *pe, *pm, *pl;
287         wcat_t    patbuf[MAXPATHLEN];
288         int rv;
289         size_t i;
290         size_t ipathc;
291         char    *path;
292
293         /* copy part up to the brace */
294         for (lm = patbuf, pm = pattern; pm != ptr; *lm++ = *pm++)
295                 ;
296         lm->w_at = 0;
297         lm->w_wc = EOS;
298         ls = lm;
299         if ((flags & GLOB_DOOFFS) == 0)
300                 gp->gl_offs = 0;
301
302         /* Find the balanced brace */
303         for (i = 0, pe = ++ptr; pe->w_wc != EOS; pe++)
304                 if (pe->w_wc == LBRACKET) {
305                         /* Ignore everything between [] */
306                         for (pm = pe++; pe->w_wc != RBRACKET &&
307                             pe->w_wc != EOS; pe++)
308                                 ;
309                         if (pe->w_wc == EOS) {
310                                 /*
311                                  * We could not find a matching RBRACKET.
312                                  * Ignore and just look for RBRACE
313                                  */
314                                 pe = pm;
315                         }
316                 } else if (pe->w_wc == LBRACE) {
317                         i++;
318                 } else if (pe->w_wc == RBRACE) {
319                         if (i == 0)
320                                 break;
321                         i--;
322                 }
323         if (!(flags & GLOB_APPEND)) {
324                 gp->gl_pathc = 0;
325                 gp->gl_pathn = gp->gl_offs + INITIAL;
326                 gp->gl_pathv = (char **)malloc(sizeof (char *) * gp->gl_pathn);
327
328         /* Non matching braces; just glob the pattern */
329         if (i != 0 || pe->w_wc == EOS)
330                 return (glob0(patbuf, pglob, limitp, errfunc));
331                 if (gp->gl_pathv == NULLCPP)
332                         return (GLOB_NOSPACE);
333                 gp->gl_pathp = gp->gl_pathv + gp->gl_offs;
```

```
311         for (i = 0, pl = pm = ptr; pm <= pe; pm++) {
312                 switch (pm->w_wc) {
313                 case LBRACKET:
314                         /* Ignore everything between [] */
315                         for (pl = pm++; pm->w_wc != RBRACKET && pm->w_wc != EOS;
316                             pm++)
317                                 ;
318                         if (pm->w_wc == EOS) {
319                                 /*
320                                  * We could not find a matching RBRACKET.
321                                  * Ignore and just look for RBRACE
322                                  */
323                                 pm = pl;
117                 for (i = 0; i < gp->gl_offs; ++i)
118                         gp->gl_pathv[i] = NULL;
324                         }
325                         break;

327                 case LBRACE:
328                         i++;
329                         break;
121         if ((path = malloc(strlen(pattern)+1)) == NULL)
122                 return (GLOB_NOSPACE);

331                 case RBRACE:
332                         if (i) {
333                                 i--;
334                                 break;
335                         }
336                         /* FALLTHROUGH */
337                 case COMMA:
338                         if (i && pm->w_wc == COMMA)
339                                 break;
340                         else {
341                                 /* Append the current string */
342                                 for (lm = ls; (pl < pm); *lm++ = *pl++)
343                                         ;
124         ipathc = gp->gl_pathc;
125         rv = globit(0, pattern, gp, flags, errfn, &path);

127         if (rv == GLOB_ABORTED) {
345                                 /*
346                                  * Append the rest of the pattern after the
347                                  * closing brace
129                  * User's error function returned non-zero, or GLOB_ERR was
130                  * set, and we encountered a directory we couldn't search.
348                                  */
349                                 for (pl = pe + 1;
350                                     (*lm++ = *pl++).w_wc != EOS; /* */)
351                                         ;

353                                 /* Expand the current pattern */
354                                 rv = globexp1(patbuf, pglob, limitp, errfunc);
355                                 if (rv && rv != GLOB_NOMATCH)
356                                         return (rv);

358                                 /* move after the comma, to the next string */
359                                 pl = pm + 1;
132                 free(path);
133                 return (GLOB_ABORTED);
360                         }
361                         break;

363                 default:
364                         break;
```



```
136         i = gp->gl_pathc - ipathc;
137         if (i >= 1 && !(flags & GLOB_NOSORT)) {
138                 qsort((char *)(gp->gl_pathp+ipathc), i, sizeof (char *),
139                     pstrcmp);
365                 }
366         }
367         return (0);
368 }



372 /*
373  * expand tilde from the passwd file.
374  */
375 static const wcat_t *
376 globtilde(const wcat_t *pattern, wcat_t *patbuf, size_t patbuf_len,
377     glob_t *pglob)
378 {
379         struct passwd *pwd;
380         char *h;
381         const wcat_t *p;
382         wcat_t *b, *eb, *q;
383         size_t n;
384         wchar_t c;

386         if (pattern->w_wc != TILDE || !(pglob->gl_flags & GLOB_TILDE))
387                 return (pattern);

389         /* Copy up to the end of the string or / */
390         eb = &patbuf[patbuf_len - 1];
391         for (p = pattern + 1, q = patbuf;
392             q < eb && p->w_wc != EOS && p->w_wc != SLASH; *q++ = *p++)
393                 ;

395         q->w_at = 0;
396         q->w_wc = EOS;

398         /* What to do if patbuf is full? */

400         if (patbuf[0].w_wc == EOS) {
401                 /*
402                  * handle a plain ~ or ~/ by expanding $HOME
403                  * first and then trying the password file
404                  */
405                 if (issetugid() != 0)
406                         return (pattern);
407                 if ((h = getenv("HOME")) == NULL) {
408                         if ((pwd = getpwuid(getuid())) == NULL)
409                                 return (pattern);
141         if (i == 0) {
142                 if (flags & GLOB_NOCHECK)
143                         (void) append(gp, pattern);
410                         else
411                                 h = pwd->pw_dir;
145                         rv = GLOB_NOMATCH;
412                 }
413         } else {
414                 /*
415                  * Expand a ~user
416                  */
417                 if ((pwd = getpwnam((char *)patbuf)) == NULL)
418                         return (pattern);
419                 else
420                         h = pwd->pw_dir;
421         }
147         gp->gl_pathp[gp->gl_pathc] = NULL;
```

```
148         free(path);

423         /* Copy the home directory */
424         for (b = patbuf; b < eb && *h != EOS; b++) {
425                 if ((n = mbtowc(&c, h, MB_LEN_MAX)) > 0) {
426                         h += n;
427                         b->w_at = 0;
428                         b->w_wc = c;
429                 } else if (n < 0) {
430                         return (pattern);
431                 } else {
432                         break;
433                 }
434         }

436         /* Append the rest of the pattern */
437         while (b < eb && (*b++ = *p++).w_wc != EOS)
438                 ;
439         b->w_at = 0;
440         b->w_wc = EOS;

442         return (patbuf);
150         return (rv);
443 }

445 static int
446 g_charclass(const wcat_t **patternp, wcat_t **bufnextp)
447 {
448         const wcat_t *pattern = *patternp + 1;
449         wcat_t *bufnext = *bufnextp;
450         const wcat_t *colon;
451         char cbuf[MB_LEN_MAX + 32];
452         wctype_t cc;
453         size_t len;

455         if ((colon = g_strchr(pattern, COLON)) == NULL ||
456             colon[1].w_wc != RBRACKET)
457                 return (1);     /* not a character class */

459         len = (size_t)(colon - pattern);
460         if (len + MB_LEN_MAX + 1 > sizeof (cbuf))
461                 return (-1);    /* invalid character class */
462         {
463                 wchar_t w;
464                 const wcat_t *s1 = pattern;
465                 char *s2 = cbuf;
466                 size_t n = len;

468                 /* Copy the string. */
469                 while (n > 0) {
470                         w = (s1++)->w_wc;
471                         /* Character class names must be ASCII. */
472                         if (iswascii(w)) {
473                                 n--;
474                                 *s2++ = w;
475                         } else {
476                                 return (-1);    /* invalid character class */
477                         }
478                 }
479                 *s2 = EOS;
480         }
481         if ((cc = wctype(cbuf)) == 0)
482                 return (-1);    /* invalid character class */
483         bufnext->w_at = M_QUOTE;
484         (bufnext++)->w_wc = M_CLASS;
485         bufnext->w_at = 0;
```



```
486         (bufnext++)->w_wc = cc;
487         *bufnextp = bufnext;
488         *patternp += len + 3;

490         return (0);
491 }

493 /*
494  * The main glob() routine: compiles the pattern (optionally processing
495  * quotes), calls glob1() to do the real pattern matching, and finally
496  * sorts the list (unless unsorted operation is requested).  Returns 0
497  * if things went well, nonzero if errors occurred.  It is not an error
498  * to find no matches.
155  * Recursive routine to match glob pattern, and walk directories.
499  */
500 static int
501 glob0(const wcat_t *pattern, glob_t *pglob, struct glob_lim *limitp,
502     int (*errfunc)(const char *, int))
157 int
158 globit(size_t dend, const char *sp, glob_t *gp, int flags,
159         int (*errfn)(const char *, int), char **path)
503 {
504         const wcat_t *qpatnext;
505         int err, oldpathc;
506         wchar_t c;
507         int a;
508         wcat_t *bufnext, patbuf[MAXPATHLEN];
161         size_t n;
162         size_t m;
163         ssize_t end = 0;        /* end of expanded directory */
164         char *pat = (char *)sp; /* pattern component */
165         char *dp = (*path) + dend;
166         int expand = 0;         /* path has pattern */
167         char *cp;
168         struct stat64 sb;
169         DIR *dirp;
170         struct dirent64 *d;
171         int err;

510         qpatnext = globtilde(pattern, patbuf, MAXPATHLEN, pglob);
511         oldpathc = pglob->gl_pathc;
512         bufnext = patbuf;

514         /*
515          * We don't need to check for buffer overflow any more.
516          * The pattern has already been copied to an internal buffer.
517          */
518         while ((a = qpatnext->w_at), (c = (qpatnext++)->w_wc) != EOS) {
519                 switch (c) {
520                 case LBRACKET:
521                         if (a != 0) {
522                                 bufnext->w_at = a;
523                                 (bufnext++)->w_wc = c;
524                                 break;
173         for (;;)
174                 switch (*dp++ = *(unsigned char *)sp++) {
175                 case '\0':      /* end of source path */
176                         if (expand)
177                                 goto Expand;
178                         else {
179                                 if (!(flags & GLOB_NOCHECK) ||
180                                     flags & (GLOB__CHECK|GLOB_MARK))
181                                         if (stat64(*path, &sb) < 0) {
182                                                 return (0);
525                         }
526                         a = qpatnext->w_at;
```

```
527                     c = qpatnext->w_wc;
528                     if (a == 0 && c == NOT)
529                             ++qpatnext;
530                     if (qpatnext->w_wc == EOS ||
531                         g_strchr(qpatnext+1, RBRACKET) == NULL) {
532                             bufnext->w_at = 0;
533                             (bufnext++)->w_wc = LBRACKET;
534                             if (a == 0 && c == NOT)
535                                     --qpatnext;
536                             break;
184                             if (flags & GLOB_MARK && S_ISDIR(sb.st_mode)) {
185                                     *dp = '\0';
186                                     *--dp = '/';
537                     }
538                     bufnext->w_at = M_QUOTE;
539                     (bufnext++)->w_wc = M_SET;
540                     if (a == 0 && c == NOT) {
541                             bufnext->w_at = M_QUOTE;
542                             (bufnext++)->w_wc = M_NOT;
543                     }
544                     a = qpatnext->w_at;
545                     c = (qpatnext++)->w_wc;
546                     do {
547                             if (a == 0 && c == LBRACKET &&
548                                 qpatnext->w_wc == COLON) {
549                                     do {
550                                             err = g_charclass(&qpatnext,
551                                                 &bufnext);
552                                             if (err)
553                                                     break;
554                                             a = qpatnext->w_at;
555                                             c = (qpatnext++)->w_wc;
556                                     } while (a == 0 && c == LBRACKET &&
557                                         qpatnext->w_wc == COLON);
558                                     if (err == -1 &&
559                                         !(pglob->gl_flags & GLOB_NOCHECK))
560                                             return (GLOB_NOMATCH);
561                                     if (a == 0 && c == RBRACKET)
562                                             break;
563                             }
564                             bufnext->w_at = a;
565                             (bufnext++)->w_wc = c;
566                             if (qpatnext->w_at == 0 &&
567                                 qpatnext->w_wc == RANGE) {
568                                     a = qpatnext[1].w_at;
569                                     c = qpatnext[1].w_wc;
570                                     if (qpatnext[1].w_at != 0 ||
571                                         qpatnext[1].w_wc != RBRACKET) {
572                                             bufnext->w_at = M_QUOTE;
573                                             (bufnext++)->w_wc = M_RNG;
574                                             bufnext->w_at = a;
575                                             (bufnext++)->w_wc = c;
576                                             qpatnext += 2;
577                                     }
578                             }
579                             a = qpatnext->w_at;
580                             c = (qpatnext++)->w_wc;
581                     } while (a != 0 || c != RBRACKET);
582                     pglob->gl_flags |= GLOB_MAGCHAR;
583                     bufnext->w_at = M_QUOTE;
584                     (bufnext++)->w_wc = M_END;
585                     break;
586             case QUESTION:
587                     if (a != 0) {
588                             bufnext->w_at = a;
589                             (bufnext++)->w_wc = c;
```



```
590                             break;
591                     }
592                     pglob->gl_flags |= GLOB_MAGCHAR;
593                     bufnext->w_at = M_QUOTE;
594                     (bufnext++)->w_wc = M_ONE;
595                     break;
596             case STAR:
597                     if (a != 0) {
598                             bufnext->w_at = a;
599                             (bufnext++)->w_wc = c;
600                             break;
601                     }
602                     pglob->gl_flags |= GLOB_MAGCHAR;
603                     /*
604                      * collapse adjacent stars to one,
605                      * to avoid exponential behavior
606                      */
607                     if (bufnext == patbuf ||
608                         bufnext[-1].w_at != M_QUOTE ||
609                         bufnext[-1].w_wc != M_ALL) {
610                             bufnext->w_at = M_QUOTE;
611                             (bufnext++)->w_wc = M_ALL;
612                     }
613                     break;
614             default:
615                     bufnext->w_at = a;
616                     (bufnext++)->w_wc = c;
617                     break;
618             }
619     }
620     bufnext->w_at = 0;
621     bufnext->w_wc = EOS;
622 #ifdef DEBUG
623     qprintf("glob0:glob1:patbuf", patbuf);
624 #endif

626     if ((err = glob1(patbuf, patbuf+MAXPATHLEN-1, pglob, limitp, errfunc))
627         != 0)
628             return (err);

630     /*
631      * If there was no match we are going to append the pattern
632      * if GLOB_NOCHECK was specified or if GLOB_NOMAGIC was specified
633      * and the pattern did not contain any magic characters
634      * GLOB_NOMAGIC is there just for compatibility with csh.
635      */
636     if (pglob->gl_pathc == oldpathc) {
637             if ((pglob->gl_flags & GLOB_NOCHECK) ||
638                 ((pglob->gl_flags & GLOB_NOMAGIC) &&
639                 !(pglob->gl_flags & GLOB_MAGCHAR)))
640                     return (globextend(pattern, pglob, limitp, NULL));
641             else
642                     return (GLOB_NOMATCH);
643     }
644     if (!(pglob->gl_flags & GLOB_NOSORT)) {
645             if ((pglob->gl_flags & GLOB_KEEPSTAT)) {
646                     /* Keep the paths and stat info synced during sort */
647                     struct glob_path_stat *path_stat;
648                     int i;
649                     int n = pglob->gl_pathc - oldpathc;
650                     int o = pglob->gl_offs + oldpathc;

652                     if ((path_stat = calloc(n, sizeof (*path_stat))) ==
653                         NULL)
188                             if (append(gp, *path) < 0) {
654                             return (GLOB_NOSPACE);
```

```
655                         for (i = 0; i < n; i++) {
656                                 path_stat[i].gps_path = pglob->gl_pathv[o + i];
657                                 path_stat[i].gps_stat = pglob->gl_statv[o + i];
658                         }
659                         qsort(path_stat, n, sizeof (*path_stat), compare_gps);
660                         for (i = 0; i < n; i++) {
661                                 pglob->gl_pathv[o + i] = path_stat[i].gps_path;
662                                 pglob->gl_statv[o + i] = path_stat[i].gps_stat;
663                         }
664                         free(path_stat);
665                 } else {
666                         qsort(pglob->gl_pathv + pglob->gl_offs + oldpathc,
667                             pglob->gl_pathc - oldpathc, sizeof (char *),
668                             compare);
669                 }
670         }
671         return (0);
672 }
192                         }
193                         /*NOTREACHED*/

674 static int
675 compare(const void *p, const void *q)
676 {
677         return (strcmp(*(char **)p, *(char **)q));
678 }
195                 case '*':
196                 case '?':
197                 case '[':
198                 case '\\':
199                         ++expand;
200                         break;

680 static int
681 compare_gps(const void *_p, const void *_q)
682 {
683         const struct glob_path_stat *p = (const struct glob_path_stat *)_p;
684         const struct glob_path_stat *q = (const struct glob_path_stat *)_q;
202                 case '/':
203                         if (expand)
204                                 goto Expand;
205                         end = dp - *path;
206                         pat = (char *)sp;
207                         break;

686         return (strcmp(p->gps_path, q->gps_path));
687 }

689 static int
690 glob1(wcat_t *pattern, wcat_t *pattern_last, glob_t *pglob,
691     struct glob_lim *limitp, int (*errfunc)(const char *, int))
692 {
693         wcat_t pathbuf[MAXPATHLEN];

695         /* A null pathname is invalid -- POSIX 1003.1 sect. 2.4. */
696         if (pattern->w_wc == EOS)
697                 return (0);
698         return (glob2(pathbuf, pathbuf+MAXPATHLEN-1,
699             pathbuf, pathbuf+MAXPATHLEN-1,
700             pattern, pattern_last, pglob, limitp, errfunc));
701 }

703 /*
704  * The functions glob2 and glob3 are mutually recursive; there is one level
705  * of recursion for each segment in the pattern that contains one or more
706  * meta characters.
```



```
707  */
708 static int
709 glob2(wcat_t *pathbuf, wcat_t *pathbuf_last, wcat_t *pathend,
710     wcat_t *pathend_last, wcat_t *pattern, wcat_t *pattern_last,
711     glob_t *pglob, struct glob_lim *limitp, int (*errfunc)(const char *, int))
712 {
713         struct stat sb;
714         wcat_t *p, *q;
715         int anymeta;

717         /*
718          * Loop over pattern segments until end of pattern or until
719          * segment with meta character found.
720          */
721         for (anymeta = 0; ; ) {
722                 if (pattern->w_wc == EOS) {             /* End of pattern? */
723                         pathend->w_at = 0;
724                         pathend->w_wc = EOS;

726                         if ((pglob->gl_flags & GLOB_LIMIT) &&
727                             limitp->glim_stat++ >= GLOB_LIMIT_STAT) {
728                                 errno = 0;
729                                 pathend->w_at = 0;
730                                 (pathend++)->w_wc = SEP;
731                                 pathend->w_at = 0;
732                                 pathend->w_wc = EOS;
733                                 return (GLOB_NOSPACE);
209                 Expand:
210                         /* determine directory and open it */
211                         (*path)[end] = '\0';
212                         dirp = opendir(**path == '\0' ? "." : *path);
213                         if (dirp == NULL) {
214                                 if (errfn != 0 && errfn(*path, errno) != 0 ||
215                                     flags&GLOB_ERR) {
216                                         return (GLOB_ABORTED);
734                         }
735                         if (g_lstat(pathbuf, &sb, pglob))
736                                 return (0);

738                         if (((pglob->gl_flags & GLOB_MARK) &&
739                             (pathend[-1].w_at != 0 ||
740                             pathend[-1].w_wc != SEP)) &&
741                             (S_ISDIR(sb.st_mode) ||
742                             (S_ISLNK(sb.st_mode) &&
743                             (g_stat(pathbuf, &sb, pglob) == 0) &&
744                             S_ISDIR(sb.st_mode)))) {
745                                 if (pathend+1 > pathend_last)
746                                         return (GLOB_NOSPACE);
747                                 pathend->w_at = 0;
748                                 (pathend++)->w_wc = SEP;
749                                 pathend->w_at = 0;
750                                 pathend->w_wc = EOS;
751                         }
752                         ++pglob->gl_matchc;
753                         return (globextend(pathbuf, pglob, limitp, &sb));
754                 }

756                 /* Find end of next segment, copy tentatively to pathend. */
757                 q = pathend;
758                 p = pattern;
759                 while (p->w_wc != EOS && p->w_wc != SEP) {
760                         if (ismeta(*p))
761                                 anymeta = 1;
762                         if (q+1 > pathend_last)
221                         /* extract pattern component */
222                         n = sp - pat;
```

```
223                         if ((cp = malloc(n)) == NULL) {
224                                 (void) closedir(dirp);
763                                 return (GLOB_NOSPACE);
764                         *q++ = *p++;
765                 }
227                         pat = memcpy(cp, pat, n);
228                         pat[n-1] = '\0';
229                         if (*--sp != '\0')
230                                 flags |= GLOB__CHECK;

767                 if (!anymeta) {         /* No expansion, do next segment. */
768                         pathend = q;
769                         pattern = p;
770                         while (pattern->w_wc == SEP) {
771                                 if (pathend+1 > pathend_last)
232                         /* expand path to max. expansion */
233                         n = dp - *path;
234                         *path = realloc(*path,
235                             strlen(*path) + NAME_MAX + strlen(sp) + 1);
236                         if (*path == NULL) {
237                                 (void) closedir(dirp);
238                                 free(pat);
772                                         return (GLOB_NOSPACE);
773                                 *pathend++ = *pattern++;
774                         }
775                 } else  {
776                         /* Need expansion, recurse. */
777                         return (glob3(pathbuf, pathbuf_last, pathend,
778                             pathend_last, pattern, p, pattern_last,
779                             pglob, limitp, errfunc));
780                 }
781         }
782         /* NOTREACHED */
783 }
241                         dp = (*path) + n;

785 static int
786 glob3(wcat_t *pathbuf, wcat_t *pathbuf_last, wcat_t *pathend,
787     wcat_t *pathend_last, wcat_t *pattern, wcat_t *restpattern,
788     wcat_t *restpattern_last, glob_t *pglob, struct glob_lim *limitp,
789     int (*errfunc)(const char *, int))
790 {
791         struct dirent *dp;
792         DIR *dirp;
793         int err;
794         char buf[MAXPATHLEN];

796         /*
797          * The readdirfunc declaration can't be prototyped, because it is
798          * assigned, below, to two functions which are prototyped in glob.h
799          * and dirent.h as taking pointers to differently typed opaque
800          * structures.
801          */
802         struct dirent *(*readdirfunc)(void *);

804         if (pathend > pathend_last)
805                 return (GLOB_NOSPACE);
806         pathend->w_at = 0;
807         pathend->w_wc = EOS;
808         errno = 0;

810         if ((dirp = g_opendir(pathbuf, pglob)) == NULL) {
811                 /* TODO: don't call for ENOENT or ENOTDIR? */
812                 if (errfunc) {
813                         if (g_Ctoc(pathbuf, buf, sizeof (buf)))
814                                 return (GLOB_ABORTED);
```



```
815                         if (errfunc(buf, errno) ||
816                             pglob->gl_flags & GLOB_ERR)
817                                 return (GLOB_ABORTED);
818                 }
819                 return (0);
820         }

243                         /* read directory and match entries */
822         err = 0;

824         /* Search directory for matching names. */
825         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
826                 readdirfunc = pglob->gl_readdir;
827         else
828                 readdirfunc = (struct dirent *(*)(void *))readdir;
829         while ((dp = (*readdirfunc)(dirp))) {
830                 char *sc;
831                 wcat_t *dc;
832                 size_t n;
833                 wchar_t w;

835                 if ((pglob->gl_flags & GLOB_LIMIT) &&
836                     limitp->glim_readdir++ >= GLOB_LIMIT_READDIR) {
837                         errno = 0;
838                         pathend->w_at = 0;
839                         (pathend++)->w_wc = SEP;
840                         pathend->w_at = 0;
841                         pathend->w_wc = EOS;
842                         err = GLOB_NOSPACE;
843                         break;
844                 }

846                 /* Initial DOT must be matched literally. */
847                 if (dp->d_name[0] == DOT && pattern->w_wc != DOT)
245                         while ((d = readdir64(dirp)) != NULL) {
246                                 cp = d->d_name;
247                                 if ((flags&GLOB_NOESCAPE)
248                                     ? fnmatch(pat, cp, FNM_PERIOD|FNM_NOESCAPE)
249                                     : fnmatch(pat, cp, FNM_PERIOD))
848                                 continue;
849                 dc = pathend;
850                 sc = dp->d_name;
851                 while (dc < pathend_last) {
852                         if ((n = mbtowc(&w, sc, MB_LEN_MAX)) <= 0) {
853                                 sc += 1;
854                                 dc->w_at = 0;
855                                 dc->w_wc = EOS;
856                         } else {
857                                 sc += n;
858                                 dc->w_at = 0;
859                                 dc->w_wc = w;
860                         }
861                         dc++;
862                         if (n <= 0)
863                                 break;
864                 }
865                 if (dc >= pathend_last) {
866                         dc->w_at = 0;
867                         dc->w_wc = EOS;
868                         err = GLOB_NOSPACE;
869                         break;
870                 }
871                 if (n < 0) {
872                         err = GLOB_NOMATCH;
873                         break;
874                 }
```

```
876                 if (!match(pathend, pattern, restpattern, GLOB_LIMIT_RECUR)) {
877                         pathend->w_at = 0;
878                         pathend->w_wc = EOS;
879                         continue;
880                 }
881                 err = glob2(pathbuf, pathbuf_last, --dc, pathend_last,
882                     restpattern, restpattern_last, pglob, limitp,
883                     errfunc);
884                 if (err)
252                         n = strlen(cp);
253                         (void) memcpy((*path) + end, cp, n);
254                         m = dp - *path;
255                         err = globit(end+n, sp, gp, flags, errfn, path);
256                         dp = (*path) + m;   /* globit can move path */
257                         if (err != 0)
885                         break;
886         }

888         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
889                 (*pglob->gl_closedir)(dirp);
890         else
891                 closedir(dirp);
261                 (void) closedir(dirp);
262                 free(pat);
892         return (err);
264                 }
265                 /* NOTREACHED */
893 }


896 /*
897  * Extend the gl_pathv member of a glob_t structure to accommodate a new item,
898  * add the new item, and update gl_pathc.
899  *
900  * This assumes the BSD realloc, which only copies the block when its size
901  * crosses a power-of-two boundary; for v7 realloc, this would cause quadratic
902  * behavior.
903  *
904  * Return 0 if new item added, error code if memory couldn't be allocated.
905  *
906  * Invariant of the glob_t structure:
907  *      Either gl_pathc is zero and gl_pathv is NULL; or gl_pathc > 0 and
908  *      gl_pathv points to (gl_offs + gl_pathc + 1) items.
269  * Comparison routine for two name arguments, called by qsort.
909  */
910 static int
911 globextend(const wcat_t *path, glob_t *pglob, struct glob_lim *limitp,
912     struct stat *sb)
271 int
272 pstrcmp(const void *npp1, const void *npp2)
913 {
914         char **pathv;
915         ssize_t i;
916         size_t newn, len;
917         char *copy = NULL;
918         const wcat_t *p;
919         struct stat **statv;
920         char junk[MB_LEN_MAX];
921         int n;

923         newn = 2 + pglob->gl_pathc + pglob->gl_offs;
924         if (pglob->gl_offs >= INT_MAX ||
925             pglob->gl_pathc >= INT_MAX ||
926             newn >= INT_MAX ||
927             SIZE_MAX / sizeof (*pathv) <= newn ||
```



```
928             SIZE_MAX / sizeof (*statv) <= newn) {
929         nospace:
930                 for (i = pglob->gl_offs; i < (ssize_t)(newn - 2); i++) {
931                         if (pglob->gl_pathv && pglob->gl_pathv[i])
932                                 free(pglob->gl_pathv[i]);
933                         if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
934                             pglob->gl_statv && pglob->gl_statv[i])
935                                 free(pglob->gl_statv[i]);
936                 }
937                 if (pglob->gl_pathv) {
938                         free(pglob->gl_pathv);
939                         pglob->gl_pathv = NULL;
940                 }
941                 if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
942                     pglob->gl_statv) {
943                         free(pglob->gl_statv);
944                         pglob->gl_statv = NULL;
945                 }
946                 return (GLOB_NOSPACE);
947         }

949         pathv = realloc(pglob->gl_pathv, newn * sizeof (*pathv));
950         if (pathv == NULL)
951                 goto nospace;
952         if (pglob->gl_pathv == NULL && pglob->gl_offs > 0) {
953                 /* first time around -- clear initial gl_offs items */
954                 pathv += pglob->gl_offs;
955                 for (i = pglob->gl_offs; --i >= 0; )
956                         *--pathv = NULL;
957         }
958         pglob->gl_pathv = pathv;

960         if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0) {
961                 statv = realloc(pglob->gl_statv, newn * sizeof (*statv));
962                 if (statv == NULL)
963                         goto nospace;
964                 if (pglob->gl_statv == NULL && pglob->gl_offs > 0) {
965                         /* first time around -- clear initial gl_offs items */
966                         statv += pglob->gl_offs;
967                         for (i = pglob->gl_offs; --i >= 0; )
968                                 *--statv = NULL;
969                 }
970                 pglob->gl_statv = statv;
971                 if (sb == NULL)
972                         statv[pglob->gl_offs + pglob->gl_pathc] = NULL;
973                 else {
974                         limitp->glim_malloc += sizeof (**statv);
975                         if ((pglob->gl_flags & GLOB_LIMIT) &&
976                             limitp->glim_malloc >= GLOB_LIMIT_MALLOC) {
977                                 errno = 0;
978                                 return (GLOB_NOSPACE);
979                         }
980                         if ((statv[pglob->gl_offs + pglob->gl_pathc] =
981                             malloc(sizeof (**statv))) == NULL)
982                                 goto copy_error;
983                         memcpy(statv[pglob->gl_offs + pglob->gl_pathc], sb,
984                             sizeof (*sb));
985                 }
986                 statv[pglob->gl_offs + pglob->gl_pathc + 1] = NULL;
987         }

989         len = MB_LEN_MAX;
990         p = path;
991         while ((n = wctomb(junk, p->w_wc)) > 0) {
992                 len += n;
993                 if ((p++)->w_wc == EOS)
```

```
 994                         break;
 995         }
 996         if (n < 0)
 997                 return (GLOB_NOMATCH);
 999         limitp->glim_malloc += len;
1000         if ((copy = malloc(len)) != NULL) {
1001                 if (g_Ctoc(path, copy, len)) {
1002                         free(copy);
1003                         return (GLOB_NOSPACE);
1004                 }
1005                 pathv[pglob->gl_offs + pglob->gl_pathc++] = copy;
1006         }
1007         pathv[pglob->gl_offs + pglob->gl_pathc] = NULL;
1009         if ((pglob->gl_flags & GLOB_LIMIT) &&
1010             (newn * sizeof (*pathv)) + limitp->glim_malloc >
1011             GLOB_LIMIT_MALLOC) {
1012                 errno = 0;
1013                 return (GLOB_NOSPACE);
1014         }
1015         copy_error:
1016         return (copy == NULL ? GLOB_NOSPACE : 0);
 274         return (strcoll(*(char **)npp1, *(char **)npp2));
1017 }
1020 /*
1021  * pattern matching function for filenames.  Each occurrence of the *
1022  * pattern causes a recursion level.
 278  * Add a new matched filename to the glob_t structure, increasing the
 279  * size of that array, as required.
1023  */
1024 static int
1025 match(wcat_t *name, wcat_t *pat, wcat_t *patend, int recur)
 281 int
 282 append(glob_t *gp, const char *str)
1026 {
1027         int ok, negate_range;
1028         wcat_t c, k;
 284         char *cp;
1030         if (recur-- == 0)
1031                 return (1);
 286         if ((cp = malloc(strlen(str)+1)) == NULL)
 287                 return (GLOB_NOSPACE);
 288         gp->gl_pathp[gp->gl_pathc++] = strcpy(cp, str);
1033         while (pat < patend) {
1034                 c = *pat++;
1035                 switch (c.w_wc) {
1036                 case M_ALL:
1037                         if (c.w_at != M_QUOTE) {
1038                                 k = *name++;
1039                                 if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1040                                         return (0);
1041                                 break;
 290         if ((gp->gl_pathc + gp->gl_offs) >= gp->gl_pathn) {
 291                 gp->gl_pathn *= 2;
 292                 gp->gl_pathv = (char **)realloc((void *)gp->gl_pathv,
 293                     gp->gl_pathn * sizeof (char *));
 294                 if (gp->gl_pathv == NULLCPP)
 295                         return (GLOB_NOSPACE);
 296                 gp->gl_pathp = gp->gl_pathv + gp->gl_offs;
1042                         }
1043                         while (pat < patend && pat->w_at == M_QUOTE &&
```



```
1044                             pat->w_wc == M_ALL)
1045                                 pat++;  /* eat consecutive '*' */
1046                         if (pat == patend)
1047                                 return (1);
1048                         do {
1049                                 if (match(name, pat, patend, recur))
1050                                         return (1);
1051                         } while ((name++)->w_wc != EOS);
1052                         return (0);
1053                 case M_ONE:
1054                         if (c.w_at != M_QUOTE) {
1055                                 k = *name++;
1056                                 if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1057                                         return (0);
1058                                 break;
1059                         }
1060                         if ((name++)->w_wc == EOS)
1061                                 return (0);
1062                         break;
1063                 case M_SET:
1064                         if (c.w_at != M_QUOTE) {
1065                                 k = *name++;
1066                                 if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1067                                         return (0);
1068                                 break;
1069                         }
1070                         ok = 0;
1071                         if ((k = *name++).w_wc == EOS)
1072                                 return (0);
1073                         if ((negate_range = (pat->w_at == M_QUOTE &&
1074                             pat->w_wc == M_NOT)) != 0)
1075                                 ++pat;
1076                         while (((c = *pat++).w_at != M_QUOTE) ||
1077                             c.w_wc != M_END) {
1078                                 if (c.w_at == M_QUOTE && c.w_wc == M_CLASS) {
1079                                         wcat_t cc;
1081                                         cc.w_at = pat->w_at;
1082                                         cc.w_wc = pat->w_wc;
1083                                         if (iswctype(k.w_wc, cc.w_wc))
1084                                                 ok = 1;
1085                                         ++pat;
1086                                 }
1087                                 if (pat->w_at == M_QUOTE &&
1088                                     pat->w_wc == M_RNG) {
1089                                         if (c.w_wc <= k.w_wc &&
1090                                             k.w_wc <= pat[1].w_wc)
1091                                                 ok = 1;
1092                                         pat += 2;
1093                                 } else if (c.w_wc == k.w_wc)
1094                                         ok = 1;
1095                         }
1096                         if (ok == negate_range)
1097                                 return (0);
1098                         break;
1099                 default:
1100                         k = *name++;
1101                         if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1102                                 return (0);
1103                         break;
1104                 }
1105         }
1106         return (name->w_wc == EOS);
1107 }
1109 /* Free allocated data belonging to a glob_t structure. */
```

```
1110 void
1111 globfree(glob_t *pglob)
1112 {
1113         int i;
1114         char **pp;

1116         if (pglob->gl_pathv != NULL) {
1117                 pp = pglob->gl_pathv + pglob->gl_offs;
1118                 for (i = pglob->gl_pathc; i--; ++pp)
1119                         if (*pp)
1120                                 free(*pp);
1121                 free(pglob->gl_pathv);
1122                 pglob->gl_pathv = NULL;
1123         }
1124         if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
1125             pglob->gl_statv != NULL) {
1126                 for (i = 0; i < pglob->gl_pathc; i++) {
1127                         if (pglob->gl_statv[i] != NULL)
1128                                 free(pglob->gl_statv[i]);
1129                 }
1130                 free(pglob->gl_statv);
1131                 pglob->gl_statv = NULL;
1132         }
1133 }

1135 static DIR *
1136 g_opendir(wcat_t *str, glob_t *pglob)
1137 {
1138         char buf[MAXPATHLEN];

1140         if (str->w_wc == EOS)
1141                 strlcpy(buf, ".", sizeof (buf));
1142         else {
1143                 if (g_Ctoc(str, buf, sizeof (buf)))
1144                         return (NULL);
1145         }

1147         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1148                 return ((*pglob->gl_opendir)(buf));

1150         return (opendir(buf));
1151 }

1153 static int
1154 g_lstat(wcat_t *fn, struct stat *sb, glob_t *pglob)
1155 {
1156         char buf[MAXPATHLEN];

1158         if (g_Ctoc(fn, buf, sizeof (buf)))
1159                 return (-1);
1160         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1161                 return ((*pglob->gl_lstat)(buf, sb));
1162         return (lstat(buf, sb));
1163 }

1165 static int
1166 g_stat(wcat_t *fn, struct stat *sb, glob_t *pglob)
1167 {
1168         char buf[MAXPATHLEN];

1170         if (g_Ctoc(fn, buf, sizeof (buf)))
1171                 return (-1);
1172         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1173                 return ((*pglob->gl_stat)(buf, sb));
1174         return (stat(buf, sb));
1175 }
```



```
1177 static wcat_t *
1178 g_strchr(const wcat_t *str, wchar_t ch)
1179 {
1180         do {
1181                 if (str->w_at == 0 && str->w_wc == ch)
1182                         return ((wcat_t *)str);
1183         } while ((str++)->w_wc != EOS);
1184         return (NULL);
1185 }

1187 static int
1188 g_Ctoc(const wcat_t *str, char *buf, uint_t len)
1189 {
1190         int n;
1191         wchar_t w;

1193         while (len >= MB_LEN_MAX) {
1194                 w = (str++)->w_wc;
1195                 if ((n = wctomb(buf, w)) > 0) {
1196                         len -= n;
1197                         buf += n;
1198                 }
1199                 if (n < 0)
1200                         break;
1201                 if (w == EOS)
1202                         return (0);
1203         }
1204         return (1);
1205 }

1207 #ifdef DEBUG
1208 static void
1209 qprintf(const char *str, wcat_t *s)
1210 {
1211         wcat_t *p;

1213         (void) printf("%s:\n", str);
1214         for (p = s; p->w_wc != EOS; p++)
1215                 (void) printf("%wc", p->w_wc);
1216         (void) printf("\n");
1217         for (p = s; p->w_wc != EOS; p++)
1218                 (void) printf("%c", p->w_at & M_PROTECT ? '"' : ' ');
1219         (void) printf("\n");
1220         for (p = s; p->w_wc != EOS; p++)
1221                 (void) printf("%c", ismeta(*p) ? '_' : ' ');
1222         (void) printf("\n");
1223 }
1224 #endif
```

```
**********************************************************
   17719 Mon Jan 28 14:19:17 2013
new/usr/src/man/man3c/glob.3c
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright (c) 1992, X/Open Company Limited. All Rights Reserved.
   3 .\" Portions Copyright (c) 2003, Sun Microsystems, Inc.  All Rights Reserved.
   4 .\" Portions Copyright (c) 2012, Gary Mills
   2 .\" Copyright (c) 1992, X/Open Company Limited. All Rights Reserved.  Portions C
   5 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
   6 .\" http://www.opengroup.org/bookstore/.
   7 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
   8 .\"
   9 .\" $OpenBSD: glob.3,v 1.30 2012/01/20 07:09:42 tedu Exp $
  10 .\"
  11 .\" Copyright (c) 1989, 1991, 1993, 1994
  12 .\" The Regents of the University of California.  All rights reserved.
  13 .\"
  14 .\" This code is derived from software contributed to Berkeley by
  15 .\" Guido van Rossum.
  16 .\" Redistribution and use in source and binary forms, with or without
  17 .\" modification, are permitted provided that the following conditions
  18 .\" are met:
  19 .\" 1. Redistributions of source code must retain the above copyright
  20 .\"    notice, this list of conditions and the following disclaimer.
  21 .\" 2. Redistributions in binary form must reproduce the above copyright
  22 .\"    notice, this list of conditions and the following disclaimer in the
  23 .\"    documentation and/or other materials provided with the distribution.
  24 .\" 3. Neither the name of the University nor the names of its contributors
  25 .\"    may be used to endorse or promote products derived from this software
  26 .\"    without specific prior written permission.
  27 .\"
  28 .\" THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
  29 .\" ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  30 .\" IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
  31 .\" ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
  32 .\" FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  33 .\" DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  34 .\" OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  35 .\" HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
  36 .\" LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
  37 .\" OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
  38 .\" SUCH DAMAGE.
  39 .\"
  40 .\"  This notice shall appear on any product containing this material.
  41 .\" The contents of this file are subject to the terms of the Common Development
  42 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
  43 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
  44 .TH GLOB 3C "Nov 1, 2003"
  45 .SH NAME
  46 glob, globfree \- generate path names matching a pattern
  47 .SH SYNOPSIS
  48 .LP
  49 .nf
  50 #include <glob.h>

  52 \fBint\fR \fBglob\fR(\fBconst char *restrict\fR \fIpattern\fR, \fBint\fR \fIflag
  53      \fBint(*\fR\fIerrfunc\fR)(const char *\fIepath\fR, int \fIeerrno)\fR,
  54      \fBglob_t *restrict\fR \fIpglob\fR);
  55 .fi

  57 .LP
  58 .nf
  59 \fBvoid\fR \fBglobfree\fR(\fBglob_t *\fR\fIpglob\fR);
```



```
  60 .fi

  62 .SH DESCRIPTION
  63 .sp
  64 .LP
  65 The \fBglob()\fR function is a path name generator.
  66 .sp
  67 .LP
  68 The \fBglobfree()\fR function frees any memory allocated by \fBglob()\fR
  69 associated with \fIpglob\fR.
  70 .SS "\fIpattern\fR Argument"
  71 .sp
  72 .LP
  73 The argument \fIpattern\fR is a pointer to a path name pattern to be expanded.
  74 The \fBglob()\fR function matches all accessible path names against this
  75 pattern and develops a list of all path names that match. In order to have
  76 access to a path name, \fBglob()\fR requires search permission on every
  77 component of a path except the last, and read permission on each directory of
  78 any filename component of \fIpattern\fR that contains any of the following
  79 special characters:
  80 .sp
  81 .in +2
  82 .nf
  83 *        ?        [
  84 .fi
  85 .in -2

  87 .SS "\fIpglob\fR Argument"
  88 .sp
  89 .LP
  90 The structure type \fBglob_t\fR is defined in the header \fB<glob.h>\fR and
  91 includes at least the following members:
  92 .sp
  93 .in +2
  94 .nf
  95 size_t   gl_pathc;     /* Total count of paths matched by */
  61 size_t   gl_pathc;     /* count of paths matched by */
  96                        /* pattern */
  97 char     **gl_pathv;   /* List of matched path names */
  98 size_t   gl_offs;      /* # of slots reserved in gl_pathv */
  99 int      gl_matchc;    /* Count of paths matching pattern. */
 100 int      gl_flags;     /* Copy of flags parameter to glob. */
  63 char     **gl_pathv;   /* pointer to list of matched */
  64                        /* path names */
  65 size_t   gl_offs;      /* slots to reserve at beginning */
  66                        /* of gl_pathv */
 101 .fi
 102 .in -2

 104 .sp
 105 .LP
 106 The \fBglob()\fR function stores the number of matched path names into
 107 \fIpglob\(mi>\fR\fBgl_pathc\fR and a pointer to a list of pointers to path
 108 names into \fIpglob\(mi>\fR\fBgl_pathv.\fR The path names are in sort order as
 109 defined by the current setting of the  \fBLC_COLLATE\fR category. The first
 110 pointer after the last path name is a \fINULL\fR pointer. If the pattern does
 111 not match any path names, the returned number of matched paths is set to 0, and
 112 the contents of \fIpglob\(mi>\fR\fBgl_pathv\fR are implementation-dependent.
 113 .sp
 114 .LP
 115 It is the caller's responsibility to create the structure pointed to by
 116 \fIpglob\fR. The \fBglob()\fR function allocates other space as needed,
 117 including the memory pointed to by \fBgl_pathv\fR. The \fBglobfree()\fR
 118 function frees any space associated with \fIpglob\fR from a previous call to
 119 \fBglob()\fR.
 120 .SS "\fIflags\fR Argument"
```

```
121 .sp
122 .LP
123 The \fIflags\fR argument is used to control the behavior of \fBglob()\fR. The
124 value of \fIflags\fR is a bitwise inclusive \fBOR\fR of zero or more of the
125 following constants, which are defined in the header <\fBglob.h\fR>:
126 .sp
127 .ne 2
128 .na
129 \fB\fBGLOB_APPEND\fR\fR
130 .ad
131 .RS 17n
132 Append path names generated to the ones from a previous call to \fBglob()\fR.
133 .RE

135 .sp
136 .ne 2
137 .na
138 \fB\fBGLOB_DOOFFS\fR\fR
139 .ad
140 .RS 17n
141 Make use of \fIpglob\(mi>\fR\fBgl_offs\fR\fI\&.\fR If this flag is set,
142 \fIpglob\(mi>\fR\fBgl_offs\fR is used to specify how many \fINULL\fR pointers
143 to add to the beginning of \fIpglob\(mi>\fR\fBgl_pathv\fR\fI\&.\fR In other
144 words, \fIpglob\(mi>\fR\fBgl_pathv\fR will point to
145 \fIpglob\(mi>\fR\fBgl_offs\fR \fINULL\fR pointers, followed by
146 \fIpglob\(mi>\fR\fBgl_pathc\fR path name pointers, followed by a \fINULL\fR
147 pointer.
148 .RE

150 .sp
151 .ne 2
152 .na
153 \fB\fBGLOB_ERR\fR\fR
154 .ad
155 .RS 17n
156 Causes \fBglob()\fR to return when it encounters a directory that it cannot
157 open or read. Ordinarily, \fBglob()\fR continues to find matches.
158 .RE

160 .sp
161 .ne 2
162 .na
163 \fB\fBGLOB_MARK\fR\fR
164 .ad
165 .RS 17n
166 Each path name that is a directory that matches \fIpattern\fR has a slash
167 appended.
168 .RE

170 .sp
171 .ne 2
172 .na
173 \fB\fBGLOB_NOCHECK\fR\fR
174 .ad
175 .RS 17n
176 If \fIpattern\fR does not match any path name, then \fBglob()\fR returns a list
177 consisting of only \fIpattern\fR, and the number of matched path names is 1.
178 .RE

180 .sp
181 .ne 2
182 .na
183 \fB\fBGLOB_NOESCAPE\fR\fR
184 .ad
185 .RS 17n
186 Disable backslash escaping.
```



```
187 .RE

189 .sp
190 .ne 2
191 .na
192 \fB\fBGLOB_NOSORT\fR\fR
193 .ad
194 .RS 17n
195 Ordinarily, \fBglob()\fR sorts the matching path names according to the current
196 setting of the \fBLC_COLLATE\fR category.  When this flag is used the order of
197 path names returned is unspecified.
198 .RE

200 .sp
201 .ne 2
202 .na
203 \fB\fBGLOB_ALTDIRFUNC\fR\fR
204 .ad
205 .RS 17n
206 The following additional fields in the \fIpglob\fR structure
207 have been initialized with alternate functions for
208 \fBglob()\fR to use to open, read, and close directories and
209 to get stat information on names found in those directories:
210 .sp
211 .nf
212 void *(*gl_opendir)(const char *);
213 struct dirent *(*gl_readdir)(void *);
214 void (*gl_closedir)(void *);
215 int (*gl_lstat)(const char *, struct stat *);
216 int (*gl_stat)(const char *, struct stat *);
217 .fi
218 .sp
219 This extension is provided to allow programs such as
220 \fBufsrestore\fR(1M) to provide globbing from directories stored
221 on tape.
222 .RE

224 .sp
225 .ne 2
226 .na
227 \fB\fBGLOB_BRACE\fR\fR
228 .ad
229 .RS 17n
230 Pre-process the pattern string to expand `{pat,pat,...}'
231 strings like \fBcsh\fR(1).  The pattern `{}' is left unexpanded
232 for historical reasons.  (\fBcsh\fR(1) does the same thing
233 to ease typing of \fBfind\fR(1) patterns.)
234 .RE

236 .sp
237 .ne 2
238 .na
239 \fB\fBGLOB_MAGCHAR\fR\fR
240 .ad
241 .RS 17n
242 Set by the \fBglob()\fR function if the pattern included globbing
243 characters.  See the description of the usage of
244 the \fBgl_matchc\fR structure member for more details.
245 .RE

247 .sp
248 .ne 2
249 .na
250 \fB\fBGLOB_NOMAGIC\fR\fR
251 .ad
252 .RS 17n
```

```
Is the same as \fBGLOB_NOCHECK\fR but it only appends the
pattern if it does not contain any of the special characters
‘*’, ‘?’, or ‘[’.  \fBGLOB_NOMAGIC\fR is provided to
simplify implementing the historic \fBcsh\fR(1) globbing behavior
and should probably not be used anywhere else.
.RE

.sp
.ne 2
.na
\fB\fBGLOB_QUOTE\fR\fR
.ad
.RS 17n
This option has no effect and is included for backwards
compatibility with older sources.
.RE

.sp
.ne 2
.na
\fB\fBGLOB_TILDE\fR\fR
.ad
.RS 17n
Expand patterns that start with ‘~’ to user name home
directories.
.RE

.sp
.ne 2
.na
\fB\fBGLOB_LIMIT\fR\fR
.ad
.RS 17n
Limit the amount of memory used by matches to \fIARG_MAX\fR.
This option should be set for programs that can be coerced
to a denial of service attack via patterns that
expand to a very large number of matches, such as a long
string of ‘*/../*/..’.
.RE

.sp
.ne 2
.na
\fB\fBGLOB_KEEPSTAT\fR\fR
.ad
.RS 17n
Retain a copy of the \fBstat\fR(2) information retrieved for
matching paths in the gl_statv array:
.sp
.nf
struct stat **gl_statv;
.fi
.sp
This option may be used to avoid \fBlstat\fR(2) lookups in
cases where they are expensive.
.RE

.sp
.LP
The \fBGLOB_APPEND\fR flag can be used to append a new set of path names to
those found in a previous call to \fBglob()\fR. The following rules apply when
two or more calls to \fBglob()\fR are made with the same value of \fIpglob\fR
and without intervening calls to \fBglobfree()\fR:
.RS +4
.TP
1.
```



```
The first such call must not set \fBGLOB_APPEND.\fR All subsequent calls
must set it.
.RE
.RS +4
.TP
2.
All the calls must set \fBGLOB_DOOFFS,\fR or all must not set it.
.RE
.RS +4
.TP
3.
After the second call, \fIpglob\(mi>\fR\fBgl_pathv\fR points to a list
containing the following:
.RS +4
.TP
a.
Zero or more \fINULL\fR pointers, as specified by \fBGLOB_DOOFFS\fR and
\fIpglob\(mi>\fR\fBgl_offs\fR.
.RE
.RS +4
.TP
b.
Pointers to the path names that were in the \fIpglob\(mi>\fR\fBgl_pathv\fR
list before the call, in the same order as before.
.RE
.RS +4
.TP
c.
Pointers to the new path names generated by the second call, in the
specified order.
.RE
.RE
.RS +4
.TP
4.
The count returned in \fIpglob\(mi>\fR\fBgl_pathc\fR will be the total
number of path names from the two calls.
.RE
.RS +4
.TP
5.
The application can change any of the fields after a call to \fBglob()\fR.
If it does, it must reset them to the original value before a subsequent call,
using the same \fIpglob\fR value, to \fBglobfree()\fR or \fBglob()\fR with the
\fBGLOB_APPEND\fR flag.
.RE
.SS "\fIerrfunc\fR and \fIepath\fR Arguments"
.sp
.LP
If, during the search, a directory is encountered that cannot be opened or read
and \fIerrfunc\fR is not a \fINULL\fR pointer, \fBglob()\fR calls
\fB(\fR\fI*errfunc\fR\fB)\fR with two arguments:
.RS +4
.TP
1.
The \fIepath\fR argument is a pointer to the path that failed.
.RE
.RS +4
.TP
2.
The \fIeerrno\fR argument is the value of \fIerrno\fR from the failure, as
set by the \fBopendir\fR(3C), \fBreaddir\fR(3C) or \fBstat\fR(2) functions.
(Other values may be used to report other errors not explicitly documented for
those functions.)
.RE
```

```
.sp
.LP
If \fB(\fR\fI*errfunc\fR\fB)\fR is called and returns non-zero, or if the
\fBGLOB_ERR\fR flag is set in \fIflags\fR, \fBglob()\fR stops the scan and
returns \fBGLOB_ABORTED\fR after setting \fIgl_pathc\fR and \fIgl_pathv\fR in
\fIpglob\fR to reflect the paths already scanned. If \fBGLOB_ERR\fR is not set
and either \fIerrfunc\fR is a \fINULL\fR pointer or
\fB(\fR\fI*errfunc\fR\fB)\fR returns 0, the error is ignored.
.SH RETURN VALUES
The following constants are defined as error return values for \fBglob()\fR:
.sp
.LP
On successful completion, \fBglob()\fR returns zero.
In addition the fields of pglob contain the values described below:

.sp
.ne 2
.na
\fB\fBgl_pathc\fR\fR
\fB\fBGLOB_ABORTED\fR\fR
.ad
.RS 16n
Contains the total number of matched pathnames so far.
This includes other matches from previous invocations of
\fBglob()\fR if \fBGLOB_APPEND\fR was specified.
The scan was stopped because \fBGLOB_ERR\fR was set or
\fB(\fR\fI*errfunc\fR\fB)\fR returned non-zero.
.RE

.sp
.ne 2
.na
\fB\fBgl_matchc\fR\fR
\fB\fBGLOB_NOMATCH\fR\fR
.ad
.RS 16n
Contains the number of matched pathnames in the current
invocation of \fBglob()\fR.
The pattern does not match any existing path name, and \fBGLOB_NOCHECK\fR was
not set in flags.
.RE

.sp
.ne 2
.na
\fB\fBgl_flags\fR\fR
\fB\fBGLOG_NOSPACE\fR\fR
.ad
.RS 16n
Contains a copy of the flags parameter with the bit
\fBGLOB_MAGCHAR\fR set if pattern contained any of the special
characters ‘*’, ‘?’, or ‘[’, cleared if not.
An attempt to allocate memory failed.
.RE

.sp
.ne 2
.na
\fB\fBgl_pathv\fR\fR
.ad
.RS 16n
Contains a pointer to a null-terminated list of matched
pathnames.  However, if \fBgl_pathc\fR is zero, the contents of
\fBgl_pathv\fR are undefined.
.RE
```



```
.LP
If \fB(\fR\fI*errfunc\fR\fB)\fR is called and returns non-zero, or if the
\fBGLOB_ERR\fR flag is set in \fIflags\fR, \fBglob()\fR stops the scan and
returns \fBGLOB_ABORTED\fR after setting \fIgl_pathc\fR and \fIgl_pathv\fR in
\fIpglob\fR to reflect the paths already scanned. If \fBGLOB_ERR\fR is not set
and either \fIerrfunc\fR is a \fINULL\fR pointer or
\fB(\fR\fI*errfunc\fR\fB)\fR returns 0, the error is ignored.
.SH RETURN VALUES
.sp
.ne 2
.na
\fB\fBgl_statv\fR\fR
.ad
.RS 16n
If the \fBGLOB_KEEPSTAT\fR flag was set, \fBgl_statv\fR contains a
pointer to a null-terminated list of matched \fBstat\fR(2)
objects corresponding to the paths in \fBgl_pathc\fR.
.RE

.sp
.LP
If \fBglob()\fR terminates due to an error, it sets \fBerrno\fR and
returns one of the following non-zero constants. defined in <\fBglob.h\fR>:

The following values are returned by \fBglob()\fR:
.sp
.ne 2
.na
\fB\fBGLOB_ABORTED\fR\fR
\fB\fB0\fR\fR
.ad
.RS 16n
The scan was stopped because \fBGLOB_ERR\fR was set or
\fB(\fR\fI*errfunc\fR\fB)\fR returned non-zero.
.RS 12n
Successful completion. The argument \fIpglob\(mi>\fR\fBgl_pathc\fR returns the
number of matched path names and the argument \fIpglob\(mi>\fR\fBgl_pathv\fR
contains a pointer to a null-terminated list of matched and sorted path names.
However, if \fIpglob\(mi>\fR\fBgl_pathc\fR is 0, the content of
\fIpglob\(mi>\fR\fBgl_pathv\fR is undefined.
.RE

.sp
.ne 2
.na
\fB\fBGLOB_NOMATCH\fR\fR
\fB\fBnon-zero\fR\fR
.ad
.RS 16n
The pattern does not match any existing path name, and \fBGLOB_NOCHECK\fR was
not set in flags.
.RS 12n
An error has occurred. Non-zero constants are defined in <\fBglob.h\fR>. The
arguments \fIpglob\(mi>\fR\fBgl_pathc\fR and \fIpglob\(mi>\fR\fBgl_pathv\fR are
still set as defined above.
.RE

.sp
.ne 2
.na
\fB\fBGLOB_NOSPACE\fR\fR
.ad
.RS 16n
An attempt to allocate memory failed.
.RE
```

```
.sp
.ne 2
.na
\fB\fBGLOB_NOSYS\fR\fR
.ad
.RS 16n
The requested function is not supported by this version of
\fBglob()\fR.
.RE

.LP
The arguments \fIpglob\(mi>\fR\fBgl_pathc\fR and \fIpglob\(mi>\fR\fBgl_pathv\fR
specified above.
.sp
.LP
The \fBglobfree()\fR function returns no value.
.SH USAGE
.sp
.LP
This function is not provided for the purpose of enabling utilities to perform
path name expansion on their arguments, as this operation is performed by the
shell, and utilities are explicitly not expected to redo this. Instead, it is
provided for applications that need to do path name expansion on strings
obtained from other sources, such as a pattern typed by a user or read from a
file.
.sp
.LP
If a utility needs to see if a path name matches a given pattern, it can use
\fBfnmatch\fR(3C).
.sp
.LP
Note that \fBgl_pathc\fR and \fBgl_pathv\fR have meaning even if \fBglob()\fR
fails. This allows \fBglob()\fR to report partial results in the event of an
error. However, if \fBgl_pathc\fR is 0, \fBgl_pathv\fR is unspecified even if
\fBglob()\fR did not return an error.
.sp
.LP
The \fBGLOB_NOCHECK\fR option could be used when an application wants to expand
a path name if wildcards are specified, but wants to treat the pattern as just
a string otherwise.
.sp
.LP
The new path names generated by a subsequent call with \fBGLOB_APPEND\fR are
not sorted together with the previous path names. This mirrors the way that the
shell handles path name expansion when multiple expansions are done on a
command line.
.sp
.LP
Applications that need tilde and parameter expansion should use the
\fBwordexp\fR(3C) function.
.SH EXAMPLES
.LP
\fBExample 1 \fRExample of \fBglob_doofs\fR function.
.sp
.LP
One use of the \fBGLOB_DOOFFS\fR flag is by applications that build an argument
list for use with the \fBexecv()\fR, \fBexecve()\fR, or \fBexecvp()\fR
functions (see \fBexec\fR(2)). Suppose, for example, that an application wants
to do the equivalent of:

.sp
.in +2
.nf
\fBls\fR \fB-l\fR *.c
.fi
.in -2
```



```
.sp
.LP
but for some reason:

.sp
.in +2
.nf
system("ls -l *.c")
.fi
.in -2

.sp
.LP
is not acceptable. The application could obtain approximately the same result
using the sequence:

.sp
.in +2
.nf
globbuf.gl_offs = 2;
glob ("*.c", GLOB_DOOFFS, NULL, &globbuf);
globbuf.gl_pathv[0] = "ls";
globbuf.gl_pathv[1] = "-l";
execvp ("ls", &globbuf.gl_pathv[0]);
.fi
.in -2

.sp
.LP
Using the same example:

.sp
.in +2
.nf
\fBls\fR \fB-l\fR *.c *.h
.fi
.in -2

.sp
.LP
could be approximately simulated using \fBGLOB_APPEND\fR as follows:

.sp
.in +2
.nf
\fBglobbuf.gl_offs = 2;
glob ("*.c", GLOB_DOOFFS, NULL, &globbuf);
glob ("*.h", GLOB_DOOFFS|GLOB_APPEND, NULL, &globbuf);
\&.\|.\|.\fR
.fi
.in -2

.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
_
```

```
 619 Interface Stability     Standard
 620 _
 621 MT-Level        MT-Safe
 622 .TE

 624 .SH SEE ALSO
 625 .sp
 626 .LP
 627 \fBexecv\fR(2), \fBstat\fR(2), \fBfnmatch\fR(3C), \fBopendir\fR(3C),
 628 \fBreaddir\fR(3C), \fBwordexp\fR(3C), \fBattributes\fR(5), \fBstandards\fR(5)
```