```
**********************************************************
   27029 Wed Mar  6 08:38:25 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) 2013 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_***

```
**********************************************************
    7731 Wed Mar  6 08:38:26 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) 2013 Gary Mills
 */

#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

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 */
#if defined(__XOPEN_OR_POSIX) && !defined(__EXTENSIONS__)
        /* The following are internal to the legacy implementation. */
        /* following are internal to the implementation */
        char    **gl_pathp;     /* gl_pathv + gl_offs */
        int     gl_pathn;       /* # of elements allocated */
#else /* defined(__XOPEN_OR_POSIX) && !defined(__EXTENSIONS__) */
        /*
         * Overlaid non-POSIX extensions, from OpenBSD:
         * These are used internally for legacy callers but
         * contain returned values for extended callers.
         */
        union {
        char    **_gl_pathp;    /* gl_pathv + gl_offs */
        int _gl_matchc;         /* Count of paths matching pattern. */
        } _gl_pama;
#define gl_pathp _gl_pama._gl_pathp
#define gl_matchc _gl_pama._gl_matchc
        union {
        int     _gl_pathn;      /* # of elements allocated */
        int _gl_flags;          /* Copy of flags parameter to glob. */
        } _gl_pafl;
#define gl_pathn _gl_pafl._gl_pathn
#define gl_flags _gl_pafl._gl_flags
        /* End of legacy glob structure */

        /* Non-POSIX extensions, from OpenBSD: */
        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 *);
#endif /* defined(__XOPEN_OR_POSIX) && !defined(__EXTENSIONS__) */
}       glob_t;
```

```
123 /*
124  * POSIX "flags" argument to glob function.
 53  * "flags" argument to glob function.
125  */
126 #define GLOB_ERR        0x0001          /* Don't continue on directory error */
127 #define GLOB_MARK       0x0002          /* Mark directories with trailing / */
128 #define GLOB_NOSORT     0x0004          /* Don't sort pathnames */
129 #define GLOB_NOCHECK    0x0008          /* Return unquoted arg if no match */
130 #define GLOB_DOOFFS     0x0010          /* Ignore gl_offs unless set */
131 #define GLOB_APPEND     0x0020          /* Append to previous glob_t */
132 #define GLOB_NOESCAPE   0x0040          /* Backslashes do not quote M-chars */
133 #define GLOB_POSIX      0x007F          /* All POSIX flags */

135 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
136 /*
137  * Non-POSIX "flags" argument to glob function, from OpenBSD.
138  */
139 #define GLOB_BRACE      0x0080  /* Expand braces ala csh. */
140 #define GLOB_MAGCHAR    0x0100  /* Pattern had globbing characters. */
141 #define GLOB_NOMAGIC    0x0200  /* GLOB_NOCHECK without magic chars (csh). */
142 #define GLOB_QUOTE      0x0400  /* Quote special chars with \. */
143 #define GLOB_TILDE      0x0800  /* Expand tilde names from the passwd file. */
144 #define GLOB_LIMIT      0x2000  /* Limit pattern match output to ARG_MAX */
145 #define GLOB_KEEPSTAT   0x4000  /* Retain stat data for paths in gl_statv. */
146 #define GLOB_ALTDIRFUNC 0x8000  /* Use alternately specified directory funcs. */
147 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */

149 /*
150  * Error returns from "glob"
151  */
152 #define GLOB_NOSYS      (-4)            /* function not supported (XPG4) */
153 #define GLOB_NOMATCH    (-3)            /* Pattern does not match */
154 #define GLOB_NOSPACE    (-2)            /* Not enough memory */
155 #define GLOB_ABORTED    (-1)            /* GLOB_ERR set or errfunc return!=0 */
156 #define GLOB_ABEND      GLOB_ABORTED    /* backward compatibility */

158 #if defined(__STDC__)

160 #ifdef  _GLOB_LIBC
161 extern int glob(const char *_RESTRICT_KYWD, int, int(*)(const char *, int),
162                 glob_t *_RESTRICT_KYWD);
163 extern int _glob_ext(const char *_RESTRICT_KYWD, int,
164                      int(*)(const char *, int), glob_t *_RESTRICT_KYWD);
165 extern void globfree(glob_t *);
166 extern void _globfree_ext(glob_t *);
167 #else /* _GLOB_LIBC */
168 #ifdef __PRAGMA_REDEFINE_EXTNAME
169 #pragma redefine_extname        glob    _glob_ext
170 #pragma redefine_extname        globfree        _globfree_ext
171 extern int glob(const char *_RESTRICT_KYWD, int, int(*)(const char *, int),
172                 glob_t *_RESTRICT_KYWD);
173 extern void globfree(glob_t *);
174 #else /* __PRAGMA_REDEFINE_EXTNAME */
175 extern int _glob_ext(const char *_RESTRICT_KYWD, int,
176                      int(*)(const char *, int), glob_t *_RESTRICT_KYWD);
177 extern void _globfree_ext(glob_t *);
178 #define glob    _glob_ext
179 #define globfree        _globfree_ext
180 #endif /* __PRAGMA_REDEFINE_EXTNAME */
181 #endif /* _GLOB_LIBC */

183 #else /* __STDC__ */

185 #ifdef  _GLOB_LIBC
 75 #else
186 extern int glob();
```



```
187 extern int _glob_ext();
188 extern void globfree();
189 extern void _globfree_ext();
190 #else /* _GLOB_LIBC */
191 #ifdef __PRAGMA_REDEFINE_EXTNAME
192 #pragma redefine_extname        glob    _glob_ext
193 #pragma redefine_extname        globfree        _globfree_ext
194 extern int glob();
195 extern void globfree();
196 #else /* __PRAGMA_REDEFINE_EXTNAME */
197 extern int _glob_ext();
198 extern void _globfree_ext();
199 #define glob    _glob_ext
200 #define globfree        _globfree_ext
201 #endif /* __PRAGMA_REDEFINE_EXTNAME */
202 #endif /* _GLOB_LIBC */
 78 #endif

205 #endif /* __STDC__ */

207 #ifdef  __cplusplus
208 }
```

_____*unchanged_portion_omitted_*

```
**********************************************************
   54369 Wed Mar  6 08:38:27 2013
new/usr/src/lib/libc/port/mapfile-vers
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
______unchanged_portion_omitted_



2538 # There should never be more than one SUNWprivate version.
2539 # Don't add any more.  Add new private symbols to SUNWprivate_1.1

2541 SYMBOL_VERSION SUNWprivate_1.1 {
2542     global:
2543         ___Argv                 { FLAGS = NODIRECT };
2544         cfree                   { FLAGS = NODIRECT };
2545         _cswidth;
2546         __ctype_mask;
2547         __environ_lock          { FLAGS = NODIRECT };
2548         __inf_read;
2549         __inf_written;
2550         __i_size;
2551         _isnanf                 { TYPE = FUNCTION; FILTER = libm.so.2 };
2552         __iswrune;
2553         __libc_threaded;
2554         _lib_version            { FLAGS = NODIRECT };
2555         _logb                   { TYPE = FUNCTION; FILTER = libm.so.2 };
2556         _lone                   { FLAGS = NODYNSORT };
2557         _lten                   { FLAGS = NODYNSORT };
2558         _lzero                  { FLAGS = NODYNSORT };
2559         __malloc_lock;
2560         _memcmp;
2561         _memcpy                 { FLAGS = NODYNSORT };
2562         _memmove;
2563         _memset;
2564         _modff                  { TYPE = FUNCTION; FILTER = libm.so.2 };
2565         __nan_read;
2566         __nan_written;
2567         __nextwctype;
2568         __nis_debug_bind;
2569         __nis_debug_calls;
2570         __nis_debug_file;
2571         __nis_debug_rpc;
2572         __nis_prefsrv;
2573         __nis_preftype;
2574         __nis_server;
2575         _nss_default_finders;
2576         __progname              { FLAGS = NODIRECT };
2577         _smbuf;
2578         _sp;
2579         __strdupa_str           { FLAGS = NODIRECT };
2580         __strdupa_len           { FLAGS = NODIRECT };
2581         _tdb_bootstrap;
2582         __threaded;
2583         thr_probe_getfunc_addr;
2584         __trans_lower;
2585         __trans_upper;
2586         _uberdata;
2587         __xpg6                  { FLAGS = NODIRECT };

2589 $if _ELF32
2590         _dladdr                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
2591         _dladdr1                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
2592         _dlclose                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
2593         _dldump                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
```



```
2594         _dlerror                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
2595         _dlinfo                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
2596         _dlmopen                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
2597         _dlopen                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
2598         _dlsym                  { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
2599         _ld_libc                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
2600         _sys_errlist;
2601         _sys_errs;
2602         _sys_index;
2603         _sys_nerr               { FLAGS = NODYNSORT };
2604         _sys_num_err;
2605 $elif sparcv9
2606         _dladdr         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
2607         _dladdr1        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
2608         _dlclose        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
2609         _dldump         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
2610         _dlerror        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
2611         _dlinfo         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
2612         _dlmopen        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
2613         _dlopen         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
2614         _dlsym          { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
2615         _ld_libc        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
2616 $elif amd64
2617         _dladdr         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
2618         _dladdr1        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
2619         _dlamd64getunwind { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
2620         _dlclose        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
2621         _dldump         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
2622         _dlerror        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
2623         _dlinfo         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
2624         _dlmopen        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
2625         _dlopen         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
2626         _dlsym          { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
2627         _ld_libc        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
2628 $else
2629 $error unknown platform
2630 $endif

2632 $if _sparc
2633         __lyday_to_month;
2634         __mon_lengths;
2635         __yday_to_month;
2636 $endif
2637 $if i386
2638         _sse_hw;
2639 $endif

2641     protected:
2642         acctctl;
2643         allocids;
2644         _assert_c99;
2645         __assert_c99;
2646         _assfail;
2647         attr_count;
2648         attr_to_data_type;
2649         attr_to_name;
2650         attr_to_option;
2651         attr_to_xattr_view;
2652         _autofssys;
2653         _bufsync;
2654         _cladm;
2655         __class_quadruple;
2656         core_get_default_content;
2657         core_get_default_path;
2658         core_get_global_content;
2659         core_get_global_path;
```

```
2660         core_get_options;
2661         core_get_process_content;
2662         core_get_process_path;
2663         core_set_default_content;
2664         core_set_default_path;
2665         core_set_global_content;
2666         core_set_global_path;
2667         core_set_options;
2668         core_set_process_content;
2669         core_set_process_path;
2670         dbm_close_status;
2671         dbm_do_nextkey;
2672         dbm_setdefwrite;
2673         _D_cplx_div;
2674         _D_cplx_div_ix;
2675         _D_cplx_div_rx;
2676         _D_cplx_mul;
2677         defclose_r;
2678         defcntl;
2679         defcntl_r;
2680         defopen;
2681         defopen_r;
2682         defread;
2683         defread_r;
2684         _delete;
2685         _dgettext;
2686         _doprnt;
2687         _doscan;
2688         _errfp;
2689         _errxfp;
2690         exportfs;
2691         _F_cplx_div;
2692         _F_cplx_div_ix;
2693         _F_cplx_div_rx;
2694         _F_cplx_mul;
2695         __fgetwc_xpg5;
2696         __fgetws_xpg5;
2697         _findbuf;
2698         _findiop;
2699         __fini_daemon_priv;
2700         _finite;
2701         _fork1                  { FLAGS = NODYNSORT };
2702         _forkall                { FLAGS = NODYNSORT };
2703         _fpclass;
2704         _fpgetmask;
2705         _fpgetround;
2706         _fpgetsticky;
2707         _fprintf;
2708         _fpsetmask;
2709         _fpsetround;
2710         _fpsetsticky;
2711         __fputwc_xpg5;
2712         __fputws_xpg5;
2713         _ftw;
2714         _gcvt;
2715         _getarg;
2716         __getcontext;
2717         _getdents;
2718         _get_exit_frame_monitor;
2719         _getfp;
2720         _getgroupsbymember;
2721         _getlogin_r;
2722         _getsp;
2723         __gettsp;
2724         getvmusage;
2725         __getwchar_xpg5;
```



```
2726         __getwc_xpg5;
2727         _glob_ext;
2728         _globfree_ext;
2729         gtty;
2730         __idmap_flush_kcache;
2731         __idmap_reg;
2732         __idmap_unreg;
2733         __init_daemon_priv;
2734         __init_suid_priv;
2735         _insert;
2736         inst_sync;
2737         _iswctype;
2738         klpd_create;
2739         klpd_getpath;
2740         klpd_getport;
2741         klpd_getucred;
2742         klpd_register;
2743         klpd_register_id;
2744         klpd_unregister;
2745         klpd_unregister_id;
2746         _lgrp_home_fast         { FLAGS = NODYNSORT };
2747         _lgrpsys;
2748         _lltostr;
2749         _lock_clear;
2750         _lock_try;
2751         _ltzset;
2752         lwp_self;
2753         makeut;
2754         makeutx;
2755         _mbftowc;
2756         mcfiller;
2757         mntopt;
2758         modctl;
2759         modutx;
2760         msgctl64;
2761         __multi_innetgr;
2762         _mutex_destroy          { FLAGS = NODYNSORT };
2763         mutex_held;
2764         _mutex_init             { FLAGS = NODYNSORT };
2765         _mutex_unlock           { FLAGS = NODYNSORT };
2766         name_to_attr;
2767         nfs_getfh;
2768         nfssvc;
2769         _nfssys;
2770         __nis_get_environment;
2771         _nss_db_state_destr;
2772         nss_default_key2str;
2773         nss_delete;
2774         nss_endent;
2775         nss_getent;
2776         _nss_initf_group;
2777         _nss_initf_netgroup;
2778         _nss_initf_passwd;
2779         _nss_initf_shadow;
2780         nss_packed_arg_init;
2781         nss_packed_context_init;
2782         nss_packed_getkey;
2783         nss_packed_set_status;
2784         nss_search;
2785         nss_setent;
2786         _nss_XbyY_fgets;
2787         __nsw_extended_action_v1;
2788         __nsw_freeconfig_v1;
2789         __nsw_getconfig_v1;
2790         __nthreads;
2791         __openattrdirat;
```

```
2792         option_to_attr;
2793         __priv_bracket;
2794         __priv_relinquish;
2795         pset_assign_forced;
2796         pset_bind_lwp;
2797         _psignal;
2798         _pthread_setcleanupinit;
2799         __putwchar_xpg5;
2800         __putwc_xpg5;
2801         rctlctl;
2802         rctllist;
2803         _realbufend;
2804         _resume;
2805         _resume_ret;
2806         _rpcsys;
2807         _sbrk_grow_aligned;
2808         scrwidth;
2809         semctl64;
2810         _semctl64;
2811         set_setcontext_enforcement;
2812         _setbufend;
2813         __set_errno;
2814         setprojrctl;
2815         _setregid;
2816         _setreuid;
2817         setsigacthandler;
2818         shmctl64;
2819         _shmctl64;
2820         sigflag;
2821         _signal;
2822         _sigoff;
2823         _sigon;
2824         _so_accept;
2825         _so_bind;
2826         _sockconfig;
2827         _so_connect;
2828         _so_getpeername;
2829         _so_getsockname;
2830         _so_getsockopt;
2831         _so_listen;
2832         _so_recv;
2833         _so_recvfrom;
2834         _so_recvmsg;
2835         _so_send;
2836         _so_sendmsg;
2837         _so_sendto;
2838         _so_setsockopt;
2839         _so_shutdown;
2840         _so_socket;
2841         _so_socketpair;
2842         str2group;
2843         str2passwd;
2844         str2spwd;
2845         __strptime_dontzero;
2846         stty;
2847         syscall;
2848         _sysconfig;
2849         __systemcall;
2850         thr_continue_allmutators;
2851         _thr_continue_allmutators;
2852         thr_continue_mutator;
2853         _thr_continue_mutator;
2854         thr_getstate;
2855         _thr_getstate;
2856         thr_mutators_barrier;
2857         _thr_mutators_barrier;
```



```
2858         thr_probe_setup;
2859         _thr_schedctl;
2860         thr_setmutator;
2861         _thr_setmutator;
2862         thr_setstate;
2863         _thr_setstate;
2864         thr_sighndlrinfo;
2865         _thr_sighndlrinfo;
2866         _thr_slot_offset;
2867         thr_suspend_allmutators;
2868         _thr_suspend_allmutators;
2869         thr_suspend_mutator;
2870         _thr_suspend_mutator;
2871         thr_wait_mutator;
2872         _thr_wait_mutator;
2873         __tls_get_addr;
2874         tpool_create;
2875         tpool_dispatch;
2876         tpool_destroy;
2877         tpool_wait;
2878         tpool_suspend;
2879         tpool_suspended;
2880         tpool_resume;
2881         tpool_member;
2882         _ttyname_dev;
2883         _ucred_alloc;
2884         ucred_getamask;
2885         _ucred_getamask;
2886         ucred_getasid;
2887         _ucred_getasid;
2888         ucred_getatid;
2889         _ucred_getatid;
2890         ucred_getauid;
2891         _ucred_getauid;
2892         _ulltostr;
2893         _uncached_getgrgid_r;
2894         _uncached_getgrnam_r;
2895         _uncached_getpwnam_r;
2896         _uncached_getpwuid_r;
2897         __ungetwc_xpg5;
2898         _unordered;
2899         utssys;
2900         _verrfp;
2901         _verrxfp;
2902         _vwarnfp;
2903         _vwarnxfp;
2904         _warnfp;
2905         _warnxfp;
2906         __wcsftime_xpg5;
2907         __wcstok_xpg5;
2908         wdbindf;
2909         wdchkind;
2910         wddelim;
2911         _wrtchk;
2912         _xflsbuf;
2913         _xgetwidth;
2914         zone_add_datalink;
2915         zone_boot;
2916         zone_check_datalink;
2917         zone_create;
2918         zone_destroy;
2919         zone_enter;
2920         zone_getattr;
2921         zone_get_id;
2922         zone_list;
2923         zone_list_datalink;
```

```
2924         zonept;
2925         zone_remove_datalink;
2926         zone_setattr;
2927         zone_shutdown;
2928         zone_version;

2930 $if _ELF32
2931         __divdi3;
2932         _file_set;
2933         _fprintf_c89;
2934         _fscanf_c89;
2935         _fwprintf_c89;
2936         _fwscanf_c89;
2937         _imaxabs_c89;
2938         _imaxdiv_c89;
2939         __moddi3;
2940         _printf_c89;
2941         _scanf_c89;
2942         _snprintf_c89;
2943         _sprintf_c89;
2944         _sscanf_c89;
2945         _strtoimax_c89;
2946         _strtoumax_c89;
2947         _swprintf_c89;
2948         _swscanf_c89;
2949         __udivdi3;
2950         __umoddi3;
2951         _vfprintf_c89;
2952         _vfscanf_c89;
2953         _vfwprintf_c89;
2954         _vfwscanf_c89;
2955         _vprintf_c89;
2956         _vscanf_c89;
2957         _vsnprintf_c89;
2958         _vsprintf_c89;
2959         _vsscanf_c89;
2960         _vswprintf_c89;
2961         _vswscanf_c89;
2962         _vwprintf_c89;
2963         _vwscanf_c89;
2964         _wcstoimax_c89;
2965         _wcstoumax_c89;
2966         _wprintf_c89;
2967         _wscanf_c89;
2968 $endif

2970 $if _sparc
2971         _cerror;
2972         install_utrap;
2973         _install_utrap;
2974         nop;
2975         _Q_cplx_div;
2976         _Q_cplx_div_ix;
2977         _Q_cplx_div_rx;
2978         _Q_cplx_lr_div;
2979         _Q_cplx_lr_div_ix;
2980         _Q_cplx_lr_div_rx;
2981         _Q_cplx_lr_mul;
2982         _Q_cplx_mul;
2983         _QgetRD;
2984         _xregs_clrptr;
2985 $endif

2987 $if sparc32
2988         __ashldi3;
2989         __ashrdi3;
```



```
2990         _cerror64;
2991         __cmpdi2;
2992         __floatdidf;
2993         __floatdisf;
2994         __floatundidf;
2995         __floatundisf;
2996         __lshrdi3;
2997         __muldi3;
2998         __ucmpdi2;
2999 $endif

3001 $if _x86
3002         _D_cplx_lr_div;
3003         _D_cplx_lr_div_ix;
3004         _D_cplx_lr_div_rx;
3005         _F_cplx_lr_div;
3006         _F_cplx_lr_div_ix;
3007         _F_cplx_lr_div_rx;
3008         __fltrounds;
3009         sysi86;
3010         _sysi86;
3011         _X_cplx_div;
3012         _X_cplx_div_ix;
3013         _X_cplx_div_rx;
3014         _X_cplx_lr_div;
3015         _X_cplx_lr_div_ix;
3016         _X_cplx_lr_div_rx;
3017         _X_cplx_mul;
3018         __xgetRD;
3019         __xtol;
3020         __xtoll;
3021         __xtoul;
3022         __xtoull;
3023 $endif

3025 $if i386
3026         __divrem64;
3027         ___tls_get_addr;
3028         __udivrem64;
3029 $endif

3031 # The following functions should not be exported from libc,
3032 # but /lib/libm.so.2, some older versions of the Studio
3033 # compiler/debugger components, and some ancient programs
3034 # found in /usr/dist reference them.  When we no longer
3035 # care about these old and broken binary objects, these
3036 # symbols should be deleted.
3037         _brk                                    { FLAGS = NODYNSORT };
3038         _cond_broadcast                         { FLAGS = NODYNSORT };
3039         _cond_init                              { FLAGS = NODYNSORT };
3040         _cond_signal                            { FLAGS = NODYNSORT };
3041         _cond_wait                              { FLAGS = NODYNSORT };
3042         _ecvt                                   { FLAGS = NODYNSORT };
3043         _fcvt                                   { FLAGS = NODYNSORT };
3044         _getc_unlocked                          { FLAGS = NODYNSORT };
3045         _llseek                                 { FLAGS = NODYNSORT };
3046         _pthread_attr_getdetachstate            { FLAGS = NODYNSORT };
3047         _pthread_attr_getinheritsched           { FLAGS = NODYNSORT };
3048         _pthread_attr_getschedparam             { FLAGS = NODYNSORT };
3049         _pthread_attr_getschedpolicy            { FLAGS = NODYNSORT };
3050         _pthread_attr_getscope                  { FLAGS = NODYNSORT };
3051         _pthread_attr_getstackaddr              { FLAGS = NODYNSORT };
3052         _pthread_attr_getstacksize              { FLAGS = NODYNSORT };
3053         _pthread_attr_init                      { FLAGS = NODYNSORT };
3054         _pthread_condattr_getpshared            { FLAGS = NODYNSORT };
3055         _pthread_condattr_init                  { FLAGS = NODYNSORT };
```

```
3056         _pthread_cond_init                     { FLAGS = NODYNSORT };
3057         _pthread_create                        { FLAGS = NODYNSORT };
3058         _pthread_getschedparam                 { FLAGS = NODYNSORT };
3059         _pthread_join                          { FLAGS = NODYNSORT };
3060         _pthread_key_create                    { FLAGS = NODYNSORT };
3061         _pthread_mutexattr_getprioceiling      { FLAGS = NODYNSORT };
3062         _pthread_mutexattr_getprotocol         { FLAGS = NODYNSORT };
3063         _pthread_mutexattr_getpshared          { FLAGS = NODYNSORT };
3064         _pthread_mutexattr_init                { FLAGS = NODYNSORT };
3065         _pthread_mutex_getprioceiling          { FLAGS = NODYNSORT };
3066         _pthread_mutex_init                    { FLAGS = NODYNSORT };
3067         _pthread_sigmask                       { FLAGS = NODYNSORT };
3068         _rwlock_init                           { FLAGS = NODYNSORT };
3069         _rw_rdlock                             { FLAGS = NODYNSORT };
3070         _rw_unlock                             { FLAGS = NODYNSORT };
3071         _rw_wrlock                             { FLAGS = NODYNSORT };
3072         _sbrk_unlocked                         { FLAGS = NODYNSORT };
3073         _select                                { FLAGS = NODYNSORT };
3074         _sema_init                             { FLAGS = NODYNSORT };
3075         _sema_post                             { FLAGS = NODYNSORT };
3076         _sema_trywait                          { FLAGS = NODYNSORT };
3077         _sema_wait                             { FLAGS = NODYNSORT };
3078         _sysfs                                 { FLAGS = NODYNSORT };
3079         _thr_create                            { FLAGS = NODYNSORT };
3080         _thr_exit                              { FLAGS = NODYNSORT };
3081         _thr_getprio                           { FLAGS = NODYNSORT };
3082         _thr_getspecific                       { FLAGS = NODYNSORT };
3083         _thr_join                              { FLAGS = NODYNSORT };
3084         _thr_keycreate                         { FLAGS = NODYNSORT };
3085         _thr_kill                              { FLAGS = NODYNSORT };
3086         _thr_main                              { FLAGS = NODYNSORT };
3087         _thr_self                              { FLAGS = NODYNSORT };
3088         _thr_setspecific                       { FLAGS = NODYNSORT };
3089         _thr_sigsetmask                        { FLAGS = NODYNSORT };
3090         _thr_stksegment                        { FLAGS = NODYNSORT };
3091         _ungetc_unlocked                       { FLAGS = NODYNSORT };

3093     local:
3094         __imax_lldiv                           { FLAGS = NODYNSORT };
3095         _ti_thr_self                           { FLAGS = NODYNSORT };
3096         *;

3098 $if lf64
3099         _seekdir64              { FLAGS = NODYNSORT };
3100         _telldir64              { FLAGS = NODYNSORT };
3101 $endif

3103 $if _sparc
3104         __cerror                { FLAGS = NODYNSORT };
3105 $endif

3107 $if sparc32
3108         __cerror64              { FLAGS = NODYNSORT };
3109 $endif

3111 $if sparcv9
3112         __cleanup               { FLAGS = NODYNSORT };
3113 $endif

3115 $if i386
3116         _syscall6               { FLAGS = NODYNSORT };
3117         __systemcall6           { FLAGS = NODYNSORT };
3118 $endif

3120 $if amd64
3121         ___tls_get_addr         { FLAGS = NODYNSORT };
```



```
3122 $endif
3123 };
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   31470 Wed Mar  6 08:38:28 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) 2013 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.
 */

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

#pragma weak _glob = glob
#pragma weak __glob_ext = _glob_ext
#pragma weak _globfree = globfree
#pragma weak __globfree_ext = _globfree_ext

#include "lint.h"

#include <sys/param.h>
#include <sys/stat.h>

#include <ctype.h>
#include <dirent.h>
#include <errno.h>
#define _GLOB_LIBC
#include <glob.h>
#undef _GLOB_LIBC
#include <limits.h>
#include <pwd.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        '['
#define NOT             '!'
#define QUESTION        '?'
#define QUOTE           '\\'
#define RANGE           '-'
#define RBRACKET        ']'
#define SEP             '/'
#define STAR            '*'
#define TILDE           '~'
#define UNDERSCORE      '_'
#define LBRACE          '{'
#define RBRACE          '}'
#define SLASH           '/'
#define COMMA           ','
#define COLON           ':'
#define GLOB__CHECK     0x80    /* stat generated paths */

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

typedef struct wcat {
        wchar_t w_wc;
        uint_t w_at;
} wcat_t;
static int      globit(size_t, const char *, glob_t *, int,
        int (*)(const char *, int), char **);
static int      pstrcmp(const void *, const void *);
static int      append(glob_t *, const char *);

#define M_ALL           '*'     /* Plus M_QUOTE */
#define M_END           ']'     /* Plus M_QUOTE */
#define M_NOT           '!'     /* Plus M_QUOTE */
#define M_ONE           '?'     /* Plus M_QUOTE */
#define M_RNG           '-'     /* Plus M_QUOTE */
#define M_SET           '['     /* Plus M_QUOTE */
#define M_CLASS         ':'     /* Plus M_QUOTE */
#define ismeta(c)       (((c).w_at&M_QUOTE) != 0)

#define GLOB_LIMIT_MALLOC       65536
#define GLOB_LIMIT_STAT         2048
#define GLOB_LIMIT_READDIR      16384

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

struct glob_lim {
        size_t  glim_malloc;
        size_t  glim_stat;
        size_t  glim_readdir;
};

struct glob_path_stat {
        char            *gps_path;
        struct stat     *gps_stat;
};

static int       compare(const void *, const void *);
static int       compare_gps(const void *, const void *);
```



```
static int       g_Ctoc(const wcat_t *, char *, uint_t);
static int       g_lstat(wcat_t *, struct stat *, glob_t *);
static DIR      *g_opendir(wcat_t *, glob_t *);
static wcat_t   *g_strchr(const wcat_t *, wchar_t);
static int       g_stat(wcat_t *, struct stat *, glob_t *);
static int       glob_com(const char *, int, int (*)(const char *, int),
                        glob_t *);
static int       glob0(const wcat_t *, glob_t *, struct glob_lim *,
                        int (*)(const char *, int));
static int       glob1(wcat_t *, wcat_t *, glob_t *, struct glob_lim *,
                        int (*)(const char *, int));
static int       glob2(wcat_t *, wcat_t *, wcat_t *, wcat_t *, wcat_t *,
                        wcat_t *, glob_t *, struct glob_lim *,
                        int (*)(const char *, int));
static int       glob3(wcat_t *, wcat_t *, wcat_t *, wcat_t *, wcat_t *,
                        wcat_t *, wcat_t *, glob_t *, struct glob_lim *,
                        int (*)(const char *, int));
static int       globextend(const wcat_t *, glob_t *, struct glob_lim *,
                   struct stat *);
static
const wcat_t    *globtilde(const wcat_t *, wcat_t *, size_t, glob_t *);
static int       globexp1(const wcat_t *, glob_t *, struct glob_lim *,
                   int (*)(const char *, int));
static int       globexp2(const wcat_t *, const wcat_t *, glob_t *,
                   struct glob_lim *, int (*)(const char *, int));
static int       match(wcat_t *, wcat_t *, wcat_t *, int);
static void      globfree_com(glob_t *);
#ifdef DEBUG
static void      qprintf(const char *, wcat_t *);
#endif

/* glob() function with legacy glob structure */
int
glob(const char *pattern, int flags, int (*errfunc)(const char *, int),
    glob_t *pglob)
{
        /* Only POSIX flags allowed */
        if ((flags & ~GLOB_POSIX) != 0)
                return (GLOB_NOMATCH);

        return (glob_com(pattern, flags, errfunc, pglob));
}

/* Extended glob() function, selected by #pragma redefine_extname in glob.h */
int
_glob_ext(const char *pattern, int flags, int (*errfunc)(const char *, int),
    glob_t *pglob)
{
        return (glob_com(pattern, flags, errfunc, pglob));
}

static int
glob_com(const char *pattern, int flags, int (*errfunc)(const char *, int),
    glob_t *pglob)
{
        const char *patnext;
        int n;
        size_t patlen;
        wchar_t c;
        wcat_t *bufnext, *bufend, patbuf[MAXPATHLEN];
        struct glob_lim limit = { 0, 0, 0 };

        if ((patlen = strnlen(pattern, PATH_MAX)) == PATH_MAX)
                return (GLOB_NOMATCH);

        patnext = pattern;
```

```
209         if (!(flags & GLOB_APPEND)) {
210                 pglob->gl_pathc = 0;
211                 pglob->gl_pathv = NULL;
212                 if ((flags & GLOB_KEEPSTAT) != 0)
213                         pglob->gl_statv = NULL;
214                 if (!(flags & GLOB_DOOFFS))
215                         pglob->gl_offs = 0;
216         }
217         pglob->gl_flags = flags & ~GLOB_MAGCHAR;
218         pglob->gl_matchc = 0;

220         if (pglob->gl_offs >= INT_MAX || pglob->gl_pathc >= INT_MAX ||
221             pglob->gl_pathc >= INT_MAX - pglob->gl_offs - 1)
222                 return (GLOB_NOSPACE);

224         bufnext = patbuf;
225         bufend = bufnext + MAXPATHLEN - 1;
226         patlen += 1;
227         if (flags & GLOB_NOESCAPE) {
228                 while (bufnext < bufend) {
229                         if ((n = mbtowc(&c, patnext, patlen)) > 0) {
230                                 patnext += n;
231                                 patlen -= n;
232                                 bufnext->w_at = 0;
233                                 (bufnext++)->w_wc = c;
234                         } else if (n == 0) {
235                                 break;
236                         } else {
237                                 return (GLOB_NOMATCH);
238                         }
239                 }
240         } else {
241                 /* Protect the quoted characters. */
242                 while (bufnext < bufend) {
243                         if ((n = mbtowc(&c, patnext, patlen)) > 0) {
244                                 patnext += n;
245                                 patlen -= n;
246                                 if (c == QUOTE) {
247                                         n = mbtowc(&c, patnext, patlen);
248                                         if (n < 0)
249                                                 return (GLOB_NOMATCH);
250                                         if (n > 0) {
251                                                 patnext += n;
252                                                 patlen -= n;
253                                         }
254                                         if (n == 0)
255                                                 c = QUOTE;
256                                         bufnext->w_at = M_PROTECT;
257                                         (bufnext++)->w_wc = c;
258                                 } else {
259                                         bufnext->w_at = 0;
260                                         (bufnext++)->w_wc = c;
261                                 }
262                         } else if (n == 0) {
263                                 break;
264                         } else {
265                                 return (GLOB_NOMATCH);
266                         }
267                 }
268         }
269         bufnext->w_at = 0;
270         bufnext->w_wc = EOS;

272         if (flags & GLOB_BRACE)
273                 return (globexp1(patbuf, pglob, &limit, errfunc));
274         else
```



```
275                 return (glob0(patbuf, pglob, &limit, errfunc));
276 }

278 /*
279  * Expand recursively a glob {} pattern. When there is no more expansion
280  * invoke the standard globbing routine to glob the rest of the magic
281  * characters
 75  * Free all space consumed by glob.
282  */
283 static int
284 globexp1(const wcat_t *pattern, glob_t *pglob, struct glob_lim *limitp,
285     int (*errfunc)(const char *, int))
 77 void
 78 globfree(glob_t *gp)
286 {
287         const wcat_t *ptr = pattern;
 80         size_t i;

289         /* Protect a single {}, for find(1), like csh */
290         if (pattern[0].w_wc == LBRACE && pattern[1].w_wc == RBRACE &&
291             pattern[2].w_wc == EOS)
292                 return (glob0(pattern, pglob, limitp, errfunc));
 82         if (gp->gl_pathv == 0)
 83                 return;

294         if ((ptr = (const wcat_t *) g_strchr(ptr, LBRACE)) != NULL)
295                 return (globexp2(ptr, pattern, pglob, limitp, errfunc));
 85         for (i = gp->gl_offs; i < gp->gl_offs + gp->gl_pathc; ++i)
 86                 free(gp->gl_pathv[i]);
 87         free((void *)gp->gl_pathv);

297         return (glob0(pattern, pglob, limitp, errfunc));
 89         gp->gl_pathc = 0;
 90         gp->gl_pathv = NULLCPP;
298 }


301 /*
302  * Recursive brace globbing helper. Tries to expand a single brace.
303  * If it succeeds then it invokes globexp1 with the new pattern.
304  * If it fails then it tries to glob the rest of the pattern and returns.
 94  * Do filename expansion.
305  */
306 static int
307 globexp2(const wcat_t *ptr, const wcat_t *pattern, glob_t *pglob,
308     struct glob_lim *limitp, int (*errfunc)(const char *, int))
 96 int
 97 glob(const char *pattern, int flags,
 98         int (*errfn)(const char *, int), glob_t *gp)
309 {
310         int     i, rv;
311         wcat_t   *lm, *ls;
312         const wcat_t *pe, *pm, *pl;
313         wcat_t    patbuf[MAXPATHLEN];
100         int rv;
101         size_t i;
102         size_t ipathc;
103         char    *path;

315         /* copy part up to the brace */
316         for (lm = patbuf, pm = pattern; pm != ptr; *lm++ = *pm++)
317                 ;
318         lm->w_at = 0;
319         lm->w_wc = EOS;
320         ls = lm;
105         if ((flags & GLOB_DOOFFS) == 0)
```

```
106                 gp->gl_offs = 0;

322         /* Find the balanced brace */
323         for (i = 0, pe = ++ptr; pe->w_wc != EOS; pe++)
324                 if (pe->w_wc == LBRACKET) {
325                         /* Ignore everything between [] */
326                         for (pm = pe++; pe->w_wc != RBRACKET &&
327                             pe->w_wc != EOS; pe++)
328                                 ;
329                         if (pe->w_wc == EOS) {
330                                 /*
331                                  * We could not find a matching RBRACKET.
332                                  * Ignore and just look for RBRACE
333                                  */
334                                 pe = pm;
335                         }
336                 } else if (pe->w_wc == LBRACE) {
337                         i++;
338                 } else if (pe->w_wc == RBRACE) {
339                         if (i == 0)
340                                 break;
341                         i--;
342                 }
108         if (!(flags & GLOB_APPEND)) {
109                 gp->gl_pathc = 0;
110                 gp->gl_pathn = gp->gl_offs + INITIAL;
111                 gp->gl_pathv = (char **)malloc(sizeof (char *) * gp->gl_pathn);

344         /* Non matching braces; just glob the pattern */
345         if (i != 0 || pe->w_wc == EOS)
346                 return (glob0(patbuf, pglob, limitp, errfunc));
113                 if (gp->gl_pathv == NULLCPP)
114                         return (GLOB_NOSPACE);
115                 gp->gl_pathp = gp->gl_pathv + gp->gl_offs;

348         for (i = 0, pl = pm = ptr; pm <= pe; pm++) {
349                 switch (pm->w_wc) {
350                 case LBRACKET:
351                         /* Ignore everything between [] */
352                         for (pl = pm++; pm->w_wc != RBRACKET && pm->w_wc != EOS;
353                             pm++)
354                                 ;
355                         if (pm->w_wc == EOS) {
356                                 /*
357                                  * We could not find a matching RBRACKET.
358                                  * Ignore and just look for RBRACE
359                                  */
360                                 pm = pl;
117                 for (i = 0; i < gp->gl_offs; ++i)
118                         gp->gl_pathv[i] = NULL;
361                         }
362                         break;

364                 case LBRACE:
365                         i++;
366                         break;
121         if ((path = malloc(strlen(pattern)+1)) == NULL)
122                 return (GLOB_NOSPACE);

368                 case RBRACE:
369                         if (i) {
370                                 i--;
371                                 break;
372                         }
373                         /* FALLTHROUGH */
374                 case COMMA:
```



```
375                         if (i && pm->w_wc == COMMA)
376                                 break;
377                         else {
378                                 /* Append the current string */
379                                 for (lm = ls; (pl < pm); *lm++ = *pl++)
380                                         ;
124         ipathc = gp->gl_pathc;
125         rv = globit(0, pattern, gp, flags, errfn, &path);

127         if (rv == GLOB_ABORTED) {
382                                 /*
383                                  * Append the rest of the pattern after the
384                                  * 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.
385                                  */
386                                 for (pl = pe + 1;
387                                     (*lm++ = *pl++).w_wc != EOS; /* */)
388                                         ;

390                                 /* Expand the current pattern */
391                                 rv = globexp1(patbuf, pglob, limitp, errfunc);
392                                 if (rv && rv != GLOB_NOMATCH)
393                                         return (rv);

395                                 /* move after the comma, to the next string */
396                                 pl = pm + 1;
132                 free(path);
133                 return (GLOB_ABORTED);
397                         }
398                         break;

400                 default:
401                         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);
402                 }
403         }
404         return (0);
405 }



409 /*
410  * expand tilde from the passwd file.
411  */
412 static const wcat_t *
413 globtilde(const wcat_t *pattern, wcat_t *patbuf, size_t patbuf_len,
414     glob_t *pglob)
415 {
416         struct passwd *pwd;
417         char *h;
418         const wcat_t *p;
419         wcat_t *b, *eb, *q;
420         int n;
421         size_t lenh;
422         wchar_t c;

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

427         /* Copy up to the end of the string or / */
428         eb = &patbuf[patbuf_len - 1];
429         for (p = pattern + 1, q = patbuf;
```

```
430             q < eb && p->w_wc != EOS && p->w_wc != SLASH; *q++ = *p++)
431                 ;
433         q->w_at = 0;
434         q->w_wc = EOS;
436         /* What to do if patbuf is full? */
438         if (patbuf[0].w_wc == EOS) {
439                 /*
440                  * handle a plain ~ or ~/ by expanding $HOME
441                  * first and then trying the password file
442                  */
443                 if (issetugid() != 0)
444                         return (pattern);
445                 if ((h = getenv("HOME")) == NULL) {
446                         if ((pwd = getpwuid(getuid())) == NULL)
447                                 return (pattern);
141         if (i == 0) {
142                 if (flags & GLOB_NOCHECK)
143                         (void) append(gp, pattern);
448                         else
449                                 h = pwd->pw_dir;
145                         rv = GLOB_NOMATCH;
450                 }
451         } else {
452                 /*
453                  * Expand a ~user
454                  */
455                 if ((pwd = getpwnam((char *)patbuf)) == NULL)
456                         return (pattern);
457                 else
458                         h = pwd->pw_dir;
459         }
147         gp->gl_pathp[gp->gl_pathc] = NULL;
148         free(path);
461         /* Copy the home directory */
462         lenh = strlen(h) + 1;
463         for (b = patbuf; b < eb && *h != EOS; b++) {
464                 if ((n = mbtowc(&c, h, lenh)) > 0) {
465                         h += n;
466                         lenh -= n;
467                         b->w_at = 0;
468                         b->w_wc = c;
469                 } else if (n < 0) {
470                         return (pattern);
471                 } else {
472                         break;
473                 }
474         }
476         /* Append the rest of the pattern */
477         while (b < eb && (*b++ = *p++).w_wc != EOS)
478                 ;
479         b->w_at = 0;
480         b->w_wc = EOS;
482         return (patbuf);
150         return (rv);
483 }
485 static int
486 g_charclass(const wcat_t **patternp, wcat_t **bufnextp)
487 {
488         const wcat_t *pattern = *patternp + 1;
```



```
489         wcat_t *bufnext = *bufnextp;
490         const wcat_t *colon;
491         char cbuf[MB_LEN_MAX + 32];
492         wctype_t cc;
493         size_t len;
495         if ((colon = g_strchr(pattern, COLON)) == NULL ||
496             colon[1].w_wc != RBRACKET)
497                 return (1);     /* not a character class */
499         len = (size_t)(colon - pattern);
500         if (len + MB_LEN_MAX + 1 > sizeof (cbuf))
501                 return (-1);    /* invalid character class */
502         {
503                 wchar_t w;
504                 const wcat_t *s1 = pattern;
505                 char *s2 = cbuf;
506                 size_t n = len;
508                 /* Copy the string. */
509                 while (n > 0) {
510                         w = (s1++)->w_wc;
511                         /* Character class names must be ASCII. */
512                         if (iswascii(w)) {
513                                 n--;
514                                 *s2++ = w;
515                         } else {
516                                 return (-1);    /* invalid character class */
517                         }
518                 }
519                 *s2 = EOS;
520         }
521         if ((cc = wctype(cbuf)) == 0)
522                 return (-1);    /* invalid character class */
523         bufnext->w_at = M_QUOTE;
524         (bufnext++)->w_wc = M_CLASS;
525         bufnext->w_at = 0;
526         (bufnext++)->w_wc = cc;
527         *bufnextp = bufnext;
528         *patternp += len + 3;
530         return (0);
531 }
533 /*
534  * The main glob() routine: compiles the pattern (optionally processing
535  * quotes), calls glob1() to do the real pattern matching, and finally
536  * sorts the list (unless unsorted operation is requested).  Returns 0
537  * if things went well, nonzero if errors occurred.  It is not an error
538  * to find no matches.
155  * Recursive routine to match glob pattern, and walk directories.
539  */
540 static int
541 glob0(const wcat_t *pattern, glob_t *pglob, struct glob_lim *limitp,
542     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)
543 {
544         const wcat_t *qpatnext;
545         int err, oldpathc;
546         wchar_t c;
547         int a;
548         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;

550         qpatnext = globtilde(pattern, patbuf, MAXPATHLEN, pglob);
551         oldpathc = pglob->gl_pathc;
552         bufnext = patbuf;

554         /*
555          * We don't need to check for buffer overflow any more.
556          * The pattern has already been copied to an internal buffer.
557          */
558         while ((a = qpatnext->w_at), (c = (qpatnext++)->w_wc) != EOS) {
559                 switch (c) {
560                 case LBRACKET:
561                         if (a != 0) {
562                                 bufnext->w_at = a;
563                                 (bufnext++)->w_wc = c;
564                                 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);
565                         }
566                         a = qpatnext->w_at;
567                         c = qpatnext->w_wc;
568                         if (a == 0 && c == NOT)
569                                 ++qpatnext;
570                         if (qpatnext->w_wc == EOS ||
571                             g_strchr(qpatnext+1, RBRACKET) == NULL) {
572                                 bufnext->w_at = 0;
573                                 (bufnext++)->w_wc = LBRACKET;
574                                 if (a == 0 && c == NOT)
575                                         --qpatnext;
576                                 break;
184                                 if (flags & GLOB_MARK && S_ISDIR(sb.st_mode)) {
185                                         *dp = '\0';
186                                         *--dp = '/';
577                         }
578                         bufnext->w_at = M_QUOTE;
579                         (bufnext++)->w_wc = M_SET;
580                         if (a == 0 && c == NOT) {
581                                 bufnext->w_at = M_QUOTE;
582                                 (bufnext++)->w_wc = M_NOT;
583                         }
584                         a = qpatnext->w_at;
585                         c = (qpatnext++)->w_wc;
586                         do {
587                                 if (a == 0 && c == LBRACKET &&
588                                     qpatnext->w_wc == COLON) {
589                                         do {
590                                                 err = g_charclass(&qpatnext,
591                                                     &bufnext);
592                                                 if (err)
```



```
593                                                         break;
594                                                 a = qpatnext->w_at;
595                                                 c = (qpatnext++)->w_wc;
596                                         } while (a == 0 && c == LBRACKET &&
597                                             qpatnext->w_wc == COLON);
598                                         if (err == -1 &&
599                                             !(pglob->gl_flags & GLOB_NOCHECK))
600                                                 return (GLOB_NOMATCH);
601                                         if (a == 0 && c == RBRACKET)
602                                                 break;
603                                 }
604                                 bufnext->w_at = a;
605                                 (bufnext++)->w_wc = c;
606                                 if (qpatnext->w_at == 0 &&
607                                     qpatnext->w_wc == RANGE) {
608                                         a = qpatnext[1].w_at;
609                                         c = qpatnext[1].w_wc;
610                                         if (qpatnext[1].w_at != 0 ||
611                                             qpatnext[1].w_wc != RBRACKET) {
612                                                 bufnext->w_at = M_QUOTE;
613                                                 (bufnext++)->w_wc = M_RNG;
614                                                 bufnext->w_at = a;
615                                                 (bufnext++)->w_wc = c;
616                                                 qpatnext += 2;
617                                         }
618                                 }
619                                 a = qpatnext->w_at;
620                                 c = (qpatnext++)->w_wc;
621                         } while (a != 0 || c != RBRACKET);
622                         pglob->gl_flags |= GLOB_MAGCHAR;
623                         bufnext->w_at = M_QUOTE;
624                         (bufnext++)->w_wc = M_END;
625                         break;
626                 case QUESTION:
627                         if (a != 0) {
628                                 bufnext->w_at = a;
629                                 (bufnext++)->w_wc = c;
630                                 break;
631                         }
632                         pglob->gl_flags |= GLOB_MAGCHAR;
633                         bufnext->w_at = M_QUOTE;
634                         (bufnext++)->w_wc = M_ONE;
635                         break;
636                 case STAR:
637                         if (a != 0) {
638                                 bufnext->w_at = a;
639                                 (bufnext++)->w_wc = c;
640                                 break;
641                         }
642                         pglob->gl_flags |= GLOB_MAGCHAR;
643                         /*
644                          * collapse adjacent stars to one,
645                          * to avoid exponential behavior
646                          */
647                         if (bufnext == patbuf ||
648                             bufnext[-1].w_at != M_QUOTE ||
649                             bufnext[-1].w_wc != M_ALL) {
650                                 bufnext->w_at = M_QUOTE;
651                                 (bufnext++)->w_wc = M_ALL;
652                         }
653                         break;
654                 default:
655                         bufnext->w_at = a;
656                         (bufnext++)->w_wc = c;
657                         break;
658                 }
```

```
659         }
660         bufnext->w_at = 0;
661         bufnext->w_wc = EOS;
662 #ifdef DEBUG
663         qprintf("glob0:glob1:patbuf", patbuf);
664 #endif

666         if ((err = glob1(patbuf, patbuf+MAXPATHLEN-1, pglob, limitp, errfunc))
667             != 0)
668                 return (err);

670         /*
671          * If there was no match we are going to append the pattern
672          * if GLOB_NOCHECK was specified or if GLOB_NOMAGIC was specified
673          * and the pattern did not contain any magic characters
674          * GLOB_NOMAGIC is there just for compatibility with csh.
675          */
676         if (pglob->gl_pathc == oldpathc) {
677                 if ((pglob->gl_flags & GLOB_NOCHECK) ||
678                     ((pglob->gl_flags & GLOB_NOMAGIC) &&
679                     !(pglob->gl_flags & GLOB_MAGCHAR)))
680                         return (globextend(pattern, pglob, limitp, NULL));
681                 else
682                         return (GLOB_NOMATCH);
683         }
684         if (!(pglob->gl_flags & GLOB_NOSORT)) {
685                 if ((pglob->gl_flags & GLOB_KEEPSTAT)) {
686                         /* Keep the paths and stat info synced during sort */
687                         struct glob_path_stat *path_stat;
688                         int i;
689                         int n = pglob->gl_pathc - oldpathc;
690                         int o = pglob->gl_offs + oldpathc;

692                         if ((path_stat = calloc(n, sizeof (*path_stat))) ==
693                             NULL)
188                                 if (append(gp, *path) < 0) {
694                                 return (GLOB_NOSPACE);
695                         for (i = 0; i < n; i++) {
696                                 path_stat[i].gps_path = pglob->gl_pathv[o + i];
697                                 path_stat[i].gps_stat = pglob->gl_statv[o + i];
698                         }
699                         qsort(path_stat, n, sizeof (*path_stat), compare_gps);
700                         for (i = 0; i < n; i++) {
701                                 pglob->gl_pathv[o + i] = path_stat[i].gps_path;
702                                 pglob->gl_statv[o + i] = path_stat[i].gps_stat;
703                         }
704                         free(path_stat);
705                 } else {
706                         qsort(pglob->gl_pathv + pglob->gl_offs + oldpathc,
707                             pglob->gl_pathc - oldpathc, sizeof (char *),
708                             compare);
709                 }
710         }
711         return (0);
712 }
192                         }
193                         /*NOTREACHED*/

714 static int
715 compare(const void *p, const void *q)
716 {
717         return (strcmp(*(char **)p, *(char **)q));
718 }
195                 case '*':
196                 case '?':
197                 case '[':
```



```
198                 case '\\':
199                         ++expand;
200                         break;

720 static int
721 compare_gps(const void *_p, const void *_q)
722 {
723         const struct glob_path_stat *p = (const struct glob_path_stat *)_p;
724         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;

726         return (strcmp(p->gps_path, q->gps_path));
727 }

729 static int
730 glob1(wcat_t *pattern, wcat_t *pattern_last, glob_t *pglob,
731     struct glob_lim *limitp, int (*errfunc)(const char *, int))
732 {
733         wcat_t pathbuf[MAXPATHLEN];

735         /* A null pathname is invalid -- POSIX 1003.1 sect. 2.4. */
736         if (pattern->w_wc == EOS)
737                 return (0);
738         return (glob2(pathbuf, pathbuf+MAXPATHLEN-1,
739             pathbuf, pathbuf+MAXPATHLEN-1,
740             pattern, pattern_last, pglob, limitp, errfunc));
741 }

743 /*
744  * The functions glob2 and glob3 are mutually recursive; there is one level
745  * of recursion for each segment in the pattern that contains one or more
746  * meta characters.
747  */
748 static int
749 glob2(wcat_t *pathbuf, wcat_t *pathbuf_last, wcat_t *pathend,
750     wcat_t *pathend_last, wcat_t *pattern, wcat_t *pattern_last,
751     glob_t *pglob, struct glob_lim *limitp, int (*errfunc)(const char *, int))
752 {
753         struct stat sb;
754         wcat_t *p, *q;
755         int anymeta;

757         /*
758          * Loop over pattern segments until end of pattern or until
759          * segment with meta character found.
760          */
761         for (anymeta = 0; ; ) {
762                 if (pattern->w_wc == EOS) {             /* End of pattern? */
763                         pathend->w_at = 0;
764                         pathend->w_wc = EOS;

766                         if ((pglob->gl_flags & GLOB_LIMIT) &&
767                             limitp->glim_stat++ >= GLOB_LIMIT_STAT) {
768                                 errno = 0;
769                                 pathend->w_at = 0;
770                                 (pathend++)->w_wc = SEP;
771                                 pathend->w_at = 0;
772                                 pathend->w_wc = EOS;
773                                 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);
774                     }
775                     if (g_lstat(pathbuf, &sb, pglob))
776                             return (0);

778                     if (((pglob->gl_flags & GLOB_MARK) &&
779                         (pathend[-1].w_at != 0 ||
780                         pathend[-1].w_wc != SEP)) &&
781                         (S_ISDIR(sb.st_mode) ||
782                         (S_ISLNK(sb.st_mode) &&
783                         (g_stat(pathbuf, &sb, pglob) == 0) &&
784                         S_ISDIR(sb.st_mode)))) {
785                             if (pathend+1 > pathend_last)
786                                     return (GLOB_NOSPACE);
787                             pathend->w_at = 0;
788                             (pathend++)->w_wc = SEP;
789                             pathend->w_at = 0;
790                             pathend->w_wc = EOS;
791                     }
792                     ++pglob->gl_matchc;
793                     return (globextend(pathbuf, pglob, limitp, &sb));
794             }

796             /* Find end of next segment, copy tentatively to pathend. */
797             q = pathend;
798             p = pattern;
799             while (p->w_wc != EOS && p->w_wc != SEP) {
800                     if (ismeta(*p))
801                             anymeta = 1;
802                     if (q+1 > pathend_last)
221                     /* extract pattern component */
222                     n = sp - pat;
223                     if ((cp = malloc(n)) == NULL) {
224                             (void) closedir(dirp);
803                             return (GLOB_NOSPACE);
804                     *q++ = *p++;
805             }
227                     pat = memcpy(cp, pat, n);
228                     pat[n-1] = '\0';
229                     if (*--sp != '\0')
230                             flags |= GLOB__CHECK;

807             if (!anymeta) {         /* No expansion, do next segment. */
808                     pathend = q;
809                     pattern = p;
810                     while (pattern->w_wc == SEP) {
811                             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);
812                                     return (GLOB_NOSPACE);
813                             *pathend++ = *pattern++;
814                     }
815             } else  {
816                     /* Need expansion, recurse. */
817                     return (glob3(pathbuf, pathbuf_last, pathend,
818                         pathend_last, pattern, p, pattern_last,
```



```
819                             pglob, limitp, errfunc));
820                 }
821         }
822         /* NOTREACHED */
823 }
241                         dp = (*path) + n;

825 static int
826 glob3(wcat_t *pathbuf, wcat_t *pathbuf_last, wcat_t *pathend,
827     wcat_t *pathend_last, wcat_t *pattern, wcat_t *restpattern,
828     wcat_t *restpattern_last, glob_t *pglob, struct glob_lim *limitp,
829     int (*errfunc)(const char *, int))
830 {
831         struct dirent *dp;
832         DIR *dirp;
833         int err;
834         char buf[MAXPATHLEN];

836         /*
837          * The readdirfunc declaration can't be prototyped, because it is
838          * assigned, below, to two functions which are prototyped in glob.h
839          * and dirent.h as taking pointers to differently typed opaque
840          * structures.
841          */
842         struct dirent *(*readdirfunc)(void *);

844         if (pathend > pathend_last)
845                 return (GLOB_NOSPACE);
846         pathend->w_at = 0;
847         pathend->w_wc = EOS;
848         errno = 0;

850         if ((dirp = g_opendir(pathbuf, pglob)) == NULL) {
851                 /* TODO: don't call for ENOENT or ENOTDIR? */
852                 if (errfunc) {
853                         if (g_Ctoc(pathbuf, buf, sizeof (buf)))
854                                 return (GLOB_ABORTED);
855                         if (errfunc(buf, errno) ||
856                             pglob->gl_flags & GLOB_ERR)
857                                 return (GLOB_ABORTED);
858                 }
859                 return (0);
860         }

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

864         /* Search directory for matching names. */
865         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
866                 readdirfunc = pglob->gl_readdir;
867         else
868                 readdirfunc = (struct dirent *(*)(void *))readdir;
869         while ((dp = (*readdirfunc)(dirp))) {
870                 char *sc;
871                 wcat_t *dc;
872                 int n;
873                 int lensc;
874                 wchar_t w;

876                 if ((pglob->gl_flags & GLOB_LIMIT) &&
877                     limitp->glim_readdir++ >= GLOB_LIMIT_READDIR) {
878                         errno = 0;
879                         pathend->w_at = 0;
880                         (pathend++)->w_wc = SEP;
881                         pathend->w_at = 0;
882                         pathend->w_wc = EOS;
```

```
883                     err = GLOB_NOSPACE;
884                     break;
885             }

887             /* Initial DOT must be matched literally. */
888             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))
889                     continue;
890             dc = pathend;
891             sc = dp->d_name;
892             lensc = strlen(sc) + 1;
893             while (dc < pathend_last) {
894                     if ((n = mbtowc(&w, sc, lensc)) <= 0) {
895                             sc += 1;
896                             lensc -= 1;
897                             dc->w_at = 0;
898                             dc->w_wc = EOS;
899                     } else {
900                             sc += n;
901                             lensc -= n;
902                             dc->w_at = 0;
903                             dc->w_wc = w;
904                     }
905                     dc++;
906                     if (n <= 0)
907                             break;
908             }
909             if (dc >= pathend_last) {
910                     dc->w_at = 0;
911                     dc->w_wc = EOS;
912                     err = GLOB_NOSPACE;
913                     break;
914             }
915             if (n < 0) {
916                     err = GLOB_NOMATCH;
917                     break;
918             }

920             if (!match(pathend, pattern, restpattern, GLOB_LIMIT_RECUR)) {
921                     pathend->w_at = 0;
922                     pathend->w_wc = EOS;
923                     continue;
924             }
925             err = glob2(pathbuf, pathbuf_last, --dc, pathend_last,
926                 restpattern, restpattern_last, pglob, limitp,
927                 errfunc);
928             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)
929                     break;
930     }

932     if (pglob->gl_flags & GLOB_ALTDIRFUNC)
933             (*pglob->gl_closedir)(dirp);
934     else
935             (void) closedir(dirp);
262                     free(pat);
936     return (err);
```



```
937 }


940 /*
941  * Extend the gl_pathv member of a glob_t structure to accommodate a new item,
942  * add the new item, and update gl_pathc.
943  *
944  * This assumes the BSD realloc, which only copies the block when its size
945  * crosses a power-of-two boundary; for v7 realloc, this would cause quadratic
946  * behavior.
947  *
948  * Return 0 if new item added, error code if memory couldn't be allocated.
949  *
950  * Invariant of the glob_t structure:
951  *      Either gl_pathc is zero and gl_pathv is NULL; or gl_pathc > 0 and
952  *      gl_pathv points to (gl_offs + gl_pathc + 1) items.
953  */
954 static int
955 globextend(const wcat_t *path, glob_t *pglob, struct glob_lim *limitp,
956     struct stat *sb)
957 {
958         char **pathv;
959         ssize_t i;
960         size_t newn, len;
961         char *copy = NULL;
962         const wcat_t *p;
963         struct stat **statv;
964         char junk[MB_LEN_MAX];
965         int n;

967         newn = 2 + pglob->gl_pathc + pglob->gl_offs;
968         if (pglob->gl_offs >= INT_MAX ||
969             pglob->gl_pathc >= INT_MAX ||
970             newn >= INT_MAX ||
971             SIZE_MAX / sizeof (*pathv) <= newn ||
972             SIZE_MAX / sizeof (*statv) <= newn) {
973         nospace:
974                 for (i = pglob->gl_offs; i < (ssize_t)(newn - 2); i++) {
975                         if (pglob->gl_pathv && pglob->gl_pathv[i])
976                                 free(pglob->gl_pathv[i]);
977                         if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
978                             pglob->gl_statv && pglob->gl_statv[i])
979                                 free(pglob->gl_statv[i]);
980                 }
981                 if (pglob->gl_pathv) {
982                         free(pglob->gl_pathv);
983                         pglob->gl_pathv = NULL;
984                 }
985                 if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
986                     pglob->gl_statv) {
987                         free(pglob->gl_statv);
988                         pglob->gl_statv = NULL;
989                 }
990                 return (GLOB_NOSPACE);
991         }

993         pathv = realloc(pglob->gl_pathv, newn * sizeof (*pathv));
994         if (pathv == NULL)
995                 goto nospace;
996         if (pglob->gl_pathv == NULL && pglob->gl_offs > 0) {
997                 /* first time around -- clear initial gl_offs items */
998                 pathv += pglob->gl_offs;
999                 for (i = pglob->gl_offs; --i >= 0; )
1000                        *--pathv = NULL;
1001        }
1002        pglob->gl_pathv = pathv;
```

```
1004         if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0) {
1005                 statv = realloc(pglob->gl_statv, newn * sizeof (*statv));
1006                 if (statv == NULL)
1007                         goto nospace;
1008                 if (pglob->gl_statv == NULL && pglob->gl_offs > 0) {
1009                         /* first time around -- clear initial gl_offs items */
1010                         statv += pglob->gl_offs;
1011                         for (i = pglob->gl_offs; --i >= 0; )
1012                                 *--statv = NULL;
1013                 }
1014                 pglob->gl_statv = statv;
1015                 if (sb == NULL)
1016                         statv[pglob->gl_offs + pglob->gl_pathc] = NULL;
1017                 else {
1018                         limitp->glim_malloc += sizeof (**statv);
1019                         if ((pglob->gl_flags & GLOB_LIMIT) &&
1020                             limitp->glim_malloc >= GLOB_LIMIT_MALLOC) {
1021                                 errno = 0;
1022                                 return (GLOB_NOSPACE);
1023                         }
1024                         if ((statv[pglob->gl_offs + pglob->gl_pathc] =
1025                             malloc(sizeof (**statv))) == NULL)
1026                                 goto copy_error;
1027                         (void) memcpy(statv[pglob->gl_offs + pglob->gl_pathc],
1028                             sb, sizeof (*sb));
1029                 }
1030                 statv[pglob->gl_offs + pglob->gl_pathc + 1] = NULL;
1031         }

1033         len = MB_LEN_MAX;
1034         p = path;
1035         while ((n = wctomb(junk, p->w_wc)) > 0) {
1036                 len += n;
1037                 if ((p++)->w_wc == EOS)
1038                         break;
1039         }
1040         if (n < 0)
1041                 return (GLOB_NOMATCH);

1043         limitp->glim_malloc += len;
1044         if ((copy = malloc(len)) != NULL) {
1045                 if (g_Ctoc(path, copy, len)) {
1046                         free(copy);
1047                         return (GLOB_NOSPACE);
1048                 }
1049                 pathv[pglob->gl_offs + pglob->gl_pathc++] = copy;
1050         }
1051         pathv[pglob->gl_offs + pglob->gl_pathc] = NULL;

1053         if ((pglob->gl_flags & GLOB_LIMIT) &&
1054             (newn * sizeof (*pathv)) + limitp->glim_malloc >
1055             GLOB_LIMIT_MALLOC) {
1056                 errno = 0;
1057                 return (GLOB_NOSPACE);
1058         }
1059         copy_error:
1060         return (copy == NULL ? GLOB_NOSPACE : 0);
 265                 /* NOTREACHED */
1061 }


1064 /*
1065  * pattern matching function for filenames.  Each occurrence of the *
1066  * pattern causes a recursion level.
 269  * Comparison routine for two name arguments, called by qsort.
```



```
1067  */
1068 static int
1069 match(wcat_t *name, wcat_t *pat, wcat_t *patend, int recur)
 271 int
 272 pstrcmp(const void *npp1, const void *npp2)
1070 {
1071         int ok, negate_range;
1072         wcat_t c, k;

1074         if (recur-- == 0)
1075                 return (1);

1077         while (pat < patend) {
1078                 c = *pat++;
1079                 switch (c.w_wc) {
1080                 case M_ALL:
1081                         if (c.w_at != M_QUOTE) {
1082                                 k = *name++;
1083                                 if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1084                                         return (0);
1085                                 break;
1086                         }
1087                         while (pat < patend && pat->w_at == M_QUOTE &&
1088                             pat->w_wc == M_ALL)
1089                                 pat++;  /* eat consecutive '*' */
1090                         if (pat == patend)
1091                                 return (1);
1092                         do {
1093                                 if (match(name, pat, patend, recur))
1094                                         return (1);
1095                         } while ((name++)->w_wc != EOS);
1096                         return (0);
1097                 case M_ONE:
1098                         if (c.w_at != M_QUOTE) {
1099                                 k = *name++;
1100                                 if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1101                                         return (0);
1102                                 break;
1103                         }
1104                         if ((name++)->w_wc == EOS)
1105                                 return (0);
1106                         break;
1107                 case M_SET:
1108                         if (c.w_at != M_QUOTE) {
1109                                 k = *name++;
1110                                 if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1111                                         return (0);
1112                                 break;
1113                         }
1114                         ok = 0;
1115                         if ((k = *name++).w_wc == EOS)
1116                                 return (0);
1117                         if ((negate_range = (pat->w_at == M_QUOTE &&
1118                             pat->w_wc == M_NOT)) != 0)
1119                                 ++pat;
1120                         while (((c = *pat++).w_at != M_QUOTE) ||
1121                             c.w_wc != M_END) {
1122                                 if (c.w_at == M_QUOTE && c.w_wc == M_CLASS) {
1123                                         wcat_t cc;

1125                                         cc.w_at = pat->w_at;
1126                                         cc.w_wc = pat->w_wc;
1127                                         if (iswctype(k.w_wc, cc.w_wc))
1128                                                 ok = 1;
1129                                         ++pat;
1130                                 }
```

```
1131                                if (pat->w_at == M_QUOTE &&
1132                                    pat->w_wc == M_RNG) {
1133                                        if (c.w_wc <= k.w_wc &&
1134                                            k.w_wc <= pat[1].w_wc)
1135                                                ok = 1;
1136                                        pat += 2;
1137                                } else if (c.w_wc == k.w_wc)
1138                                        ok = 1;
1139                        }
1140                        if (ok == negate_range)
1141                                return (0);
1142                        break;
1143                default:
1144                        k = *name++;
1145                        if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1146                                return (0);
1147                        break;
1148                }
1149        }
1150        return (name->w_wc == EOS);
 274        return (strcoll(*(char **)npp1, *(char **)npp2));
1151 }

1153 /* globfree() function with legacy glob structure */
1154 void
1155 globfree(glob_t *pglob)
1156 {
1157         /* Only POSIX flags allowed */
1158         pglob->gl_flags &= GLOB_POSIX;

1160         globfree_com(pglob);
1161 }

1163 /*
1164  * Extended globfree() function, selected by #pragma redefine_extname
1165  * in glob.h
 278  * Add a new matched filename to the glob_t structure, increasing the
 279  * size of that array, as required.
1166  */
1167 void
1168 _globfree_ext(glob_t *pglob)
 281 int
 282 append(glob_t *gp, const char *str)
1169 {
1170         globfree_com(pglob);
1171 }
 284         char *cp;

1173 /* Free allocated data belonging to a glob_t structure. */
1174 void
1175 globfree_com(glob_t *pglob)
1176 {
1177         int i;
1178         char **pp;
 286         if ((cp = malloc(strlen(str)+1)) == NULL)
 287                 return (GLOB_NOSPACE);
 288         gp->gl_pathp[gp->gl_pathc++] = strcpy(cp, str);

1180         if (pglob->gl_pathv != NULL) {
1181                 pp = pglob->gl_pathv + pglob->gl_offs;
1182                 for (i = pglob->gl_pathc; i--; ++pp)
1183                         if (*pp)
1184                                 free(*pp);
1185                 free(pglob->gl_pathv);
1186                 pglob->gl_pathv = NULL;
 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;
1187         }
1188         if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
1189             pglob->gl_statv != NULL) {
1190                 for (i = 0; i < pglob->gl_pathc; i++) {
1191                         if (pglob->gl_statv[i] != NULL)
1192                                 free(pglob->gl_statv[i]);
1193                 }
1194                 free(pglob->gl_statv);
1195                 pglob->gl_statv = NULL;
1196         }
1197 }

1199 static DIR *
1200 g_opendir(wcat_t *str, glob_t *pglob)
1201 {
1202         char buf[MAXPATHLEN];

1204         if (str->w_wc == EOS)
1205                 (void) strlcpy(buf, ".", sizeof (buf));
1206         else {
1207                 if (g_Ctoc(str, buf, sizeof (buf)))
1208                         return (NULL);
1209         }

1211         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1212                 return ((*pglob->gl_opendir)(buf));

1214         return (opendir(buf));
1215 }

1217 static int
1218 g_lstat(wcat_t *fn, struct stat *sb, glob_t *pglob)
1219 {
1220         char buf[MAXPATHLEN];

1222         if (g_Ctoc(fn, buf, sizeof (buf)))
1223                 return (-1);
1224         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1225                 return ((*pglob->gl_lstat)(buf, sb));
1226         return (lstat(buf, sb));
1227 }

1229 static int
1230 g_stat(wcat_t *fn, struct stat *sb, glob_t *pglob)
1231 {
1232         char buf[MAXPATHLEN];

1234         if (g_Ctoc(fn, buf, sizeof (buf)))
1235                 return (-1);
1236         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1237                 return ((*pglob->gl_stat)(buf, sb));
1238         return (stat(buf, sb));
1239 }

1241 static wcat_t *
1242 g_strchr(const wcat_t *str, wchar_t ch)
1243 {
1244         do {
1245                 if (str->w_at == 0 && str->w_wc == ch)
1246                         return ((wcat_t *)str);
```

```
1247         } while ((str++)->w_wc != EOS);
1248         return (NULL);
1249 }

1251 static int
1252 g_Ctoc(const wcat_t *str, char *buf, uint_t len)
1253 {
1254         int n;
1255         wchar_t w;

1257         while (len >= MB_LEN_MAX) {
1258                 w = (str++)->w_wc;
1259                 if ((n = wctomb(buf, w)) > 0) {
1260                         len -= n;
1261                         buf += n;
1262                 }
1263                 if (n < 0)
1264                         break;
1265                 if (w == EOS)
1266                         return (0);
1267         }
1268         return (1);
1269 }

1271 #ifdef DEBUG
1272 static void
1273 qprintf(const char *str, wcat_t *s)
1274 {
1275         wcat_t *p;

1277         (void) printf("%s:\n", str);
1278         for (p = s; p->w_wc != EOS; p++)
1279                 (void) printf("%wc", p->w_wc);
1280         (void) printf("\n");
1281         for (p = s; p->w_wc != EOS; p++)
1282                 (void) printf("%c", p->w_at & M_PROTECT ? '"' : ' ');
1283         (void) printf("\n");
1284         for (p = s; p->w_wc != EOS; p++)
1285                 (void) printf("%c", ismeta(*p) ? '_' : ' ');
1286         (void) printf("\n");
1287 }
1288 #endif
```

```
**********************************************************
   17719 Wed Mar  6 08:38:29 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) 2013, 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)
```