

new/usr/src/cmd/ssh/include/config.h

1

```
*****
27029 Wed Jan 30 08:38:37 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()
*****
1 /* config.h. Generated by configure. */
2 /* config.h.in. Generated from configure.ac by autoheader. */
3 /* $Id: acconfig.h,v 1.145 2002/09/26 00:38:48 tim Exp $ */

5 /*
6 * Copyright (c) 2001, 2010, Oracle and/or its affiliates. All rights reserved.
7 * Copyright (c) 2012 Gary Mills
8 */

10 #ifndef _CONFIG_H
11 #define _CONFIG_H

13 #ifdef __cplusplus
14 extern "C" {
15 #endif

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

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

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

29 /* SCO workaround */
30 /* #undef BROKEN_SYS_TERMIO_H */

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

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

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

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

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

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

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

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

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

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

new/usr/src/cmd/ssh/include/config.h

2

```
61 #define IP_TOS_IS_BROKEN 1

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

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

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

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

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

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

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

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

87 /*
88 * Define if you have/want arrays (cluster-wide session management, not C
89 * arrays)
90 */
91 /* #undef WITH_IRIX_ARRAY */

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

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

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

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

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

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

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

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

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

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

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

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

new/usr/src/cmd/ssh/include/config.h

3

```

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

```

new/usr/src/cmd/ssh/include/config.h

4

```

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 HAVE_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 */

```

new/usr/src/cmd/ssh/include/config.h

5

```

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

```

new/usr/src/cmd/ssh/include/config.h

6

```

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 explicitly 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-gnx systems */
400 /* #undef MISSING_NFDBITS */

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

405 /* Define on *nto-gnx 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 'gettutent' function. */
532 #define HAVE_GETTUTENT 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, mkstemp 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_sysm.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_SETTUTENT 1

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


new/usr/src/head/glob.h

1

```
*****
5560 Wed Jan 30 08:38:38 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()
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License, Version 1.0 only
6  * (the "License").  You may not use this file except in compliance
7  * with the License.
8  *
9  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10 * or http://www.opensolaris.org/os/licensing.
11 * See the License for the specific language governing permissions
12 * and limitations under the License.
13 *
14 * When distributing Covered Code, include this CDDL HEADER in each
15 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16 * If applicable, add the following below this CDDL HEADER, with the
17 * fields enclosed by brackets "[" replaced with your own identifying
18 * information: Portions Copyright [yyyy] [name of copyright owner]
19 *
20 * CDDL HEADER END
21 */
23 /*
24 * Copyright (c) 1989, 1993
25 *   The Regents of the University of California.  All rights reserved.
26 *
27 * This code is derived from software contributed to Berkeley by
28 * Guido van Rossum.
29 *
30 * Redistribution and use in source and binary forms, with or without
31 * modification, are permitted provided that the following conditions
32 * are met:
33 * 1. Redistributions of source code must retain the above copyright
34 * notice, this list of conditions and the following disclaimer.
35 * 2. Redistributions in binary form must reproduce the above copyright
36 * notice, this list of conditions and the following disclaimer in the
37 * documentation and/or other materials provided with the distribution.
38 * 3. Neither the name of the University nor the names of its contributors
39 * may be used to endorse or promote products derived from this software
40 * without specific prior written permission.
41 *
42 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
43 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
44 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
45 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
46 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
47 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
48 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
49 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
50 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
51 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
52 * SUCH DAMAGE.
53 *
54 *   @(#)glob.h      8.1 (Berkeley) 6/2/93
55 */
57 /*
58 * Copyright 2003 Sun Microsystems, Inc.  All rights reserved.
59 * Use is subject to license terms.
60 * Copyright (c) 2012 Gary Mills
```

new/usr/src/head/glob.h

2

```
61 */
63 /*
64 * Copyright 1985, 1992 by Mortice Kern Systems Inc.  All rights reserved.
65 */
67 #ifndef _GLOB_H
68 #define _GLOB_H
69
70 #pragma ident      "%Z%M% %I%      %E% SMI"
71
72 #include <sys/feature_tests.h>
73 #include <sys/types.h>
74 #include <sys/stat.h>
75 #include <dirent.h>
76
77 #ifdef __cplusplus
78 extern "C" {
79 #endif
80
81 struct stat;
82
83 typedef struct glob_t {
84     /* Members required by POSIX */
85     size_t gl_pathc;      /* Total count of paths matched by pattern */
86     size_t gl_offc;      /* Count of paths matched by pattern */
87     char **gl_pathv;     /* List of matched pathnames */
88     size_t gl_offs;      /* # of slots reserved in gl_pathv */
89     /* Non-POSIX extensions, from Openbsd */
90     int gl_matchc;       /* Count of paths matching pattern. */
91     int gl_flags;        /* Copy of flags parameter to glob. */
92     /* Members only accessed when Non-POSIX flags are specified. */
93     struct stat **gl_statv; /* Stat entries corresponding to gl_pathv */
94     /*
95      * Alternate filesystem access methods for glob; replacement
96      * versions of closedir(3), readdir(3), opendir(3), stat(2)
97      * and lstat(2).
98      */
99     void (*gl_closedir)(void *);
100    struct dirent *(*gl_readdir)(void *);
101    void (*gl_opendir)(const char *);
102    int (*gl_lstat)(const char *, struct stat *);
103    int (*gl_stat)(const char *, struct stat *);
104    /* following are internal to the implementation */
105    char **gl_pathp;      /* gl_pathv + gl_offs */
106    int gl_pathn;         /* # of elements allocated */
107 } glob_t;
108
109 /*
110 * POSIX "flags" argument to glob function.
111 * "flags" argument to glob function.
112 */
113 #define GLOB_ERR      0x0001    /* Don't continue on directory error */
114 #define GLOB_MARK     0x0002    /* Mark directories with trailing / */
115 #define GLOB_NOSORT   0x0004    /* Don't sort pathnames */
116 #define GLOB_NOCHECK  0x0008    /* Return unquoted arg if no match */
117 #define GLOB_DOOFFS  0x0010    /* Ignore gl_offs unless set */
118 #define GLOB_APPEND   0x0020    /* Append to previous glob_t */
119 #define GLOB_NOESCAPE 0x0040    /* Backslashes do not quote M-chars */
120
121 /*
122 * Non-POSIX "flags" argument to glob function, from Openbsd.
123 */
124 #define GLOB_BRACE    0x0080    /* Expand braces ala csh. */
125 #define GLOB_MAGCHAR  0x0100    /* Pattern had globbing characters. */
126 #define GLOB_NOMAGIC  0x0200    /* GLOB_NOCHECK without magic chars (csh). */
```

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

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

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

144 #ifdef __cplusplus
145 }
_____unchanged_portion_omitted_
```

new/usr/src/lib/libc/port/regex/THIRDPARTYLICENSE

1

1624 Wed Jan 30 08:38:39 2013

new/usr/src/lib/libc/port/regex/THIRDPARTYLICENSE

1097 glob(3c) needs to support non-POSIX options

3341 The sftp command should use the native glob()

1 Copyright (c) 2013 Gary Mills

2 Copyright (c) 1989, 1993

3 The Regents of the University of California. All rights reserved.

5 This code is derived from software contributed to Berkeley by

6 Guido van Rossum.

8 Redistribution and use in source and binary forms, with or without

9 modification, are permitted provided that the following conditions

10 are met:

11 1. Redistributions of source code must retain the above copyright

12 notice, this list of conditions and the following disclaimer.

13 2. Redistributions in binary form must reproduce the above copyright

14 notice, this list of conditions and the following disclaimer in the

15 documentation and/or other materials provided with the distribution.

16 3. Neither the name of the University nor the names of its contributors

17 may be used to endorse or promote products derived from this software

18 without specific prior written permission.

20 THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND

21 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

22 IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

23 ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE

24 FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

25 DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS

26 OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)

27 HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT

28 LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY

29 OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF

30 SUCH DAMAGE.

new/usr/src/lib/libc/port/regex/THIRDPARTYLICENSE.descrip 1

```
*****  
25 Wed Jan 30 08:38:40 2013  
new/usr/src/lib/libc/port/regex/THIRDPARTYLICENSE.descrip  
1097 glob(3c) needs to support non-POSIX options  
3341 The sftp command should use the native glob()  
*****  
1 ALL OF THE GLOB FUNCTION
```

```

*****
30302 Wed Jan 30 08:38:41 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()
*****
1 /*
2  * Copyright (c) 2012 Gary Mills
3  */
4 /*      $OpenBSD: glob.c,v 1.39 2012/01/20 07:09:42 tedu Exp $ */
5 /*
6  * Copyright (c) 1989, 1993
7  *   The Regents of the University of California.  All rights reserved.
8  *   CDDL HEADER START
9  *
10 * This code is derived from software contributed to Berkeley by
11 * Guido van Rossum.
12 * The contents of this file are subject to the terms of the
13 * Common Development and Distribution License (the "License").
14 * You may not use this file except in compliance with the License.
15 *
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 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
28 * or http://www.opensolaris.org/os/licensing.
29 * See the License for the specific language governing permissions
30 * and limitations under the License.
31 *
32 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
33 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
34 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
35 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
37 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
38 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
39 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
40 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
41 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
42 * SUCH DAMAGE.
43 * When distributing Covered Code, include this CDDL HEADER in each
44 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
45 * If applicable, add the following below this CDDL HEADER, with the
46 * fields enclosed by brackets "[]" replaced with your own identifying
47 * information: Portions Copyright [yyyy] [name of copyright owner]
48 *
49 * CDDL HEADER END
50 */
51
52 /*
53  * glob(3) -- a superset of the one defined in POSIX 1003.2.
54  * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
55  * Use is subject to license terms.
56  */
57
58 /*
59  * This code is MKS code ported to Solaris originally with minimum
60  * modifications so that upgrades from MKS would readily integrate.

```

```

30 * The MKS basis for this modification was:
31 *
32 * The [!...] convention to negate a range is supported (SysV, Posix, ksh).
33 *   $Id: glob.c 1.31 1994/04/07 22:50:43 mark
34 *
35 * Optional extra services, controlled by flags not defined by POSIX:
36 * Additional modifications have been made to this code to make it
37 * 64-bit clean.
38 */
39
40 * glob, globfree -- POSIX.2 compatible file name expansion routines.
41 *
42 * GLOB_QUOTE:
43 * Escaping convention: \ inhibits any special meaning the following
44 * character might have (except \ at end of string is retained).
45 * GLOB_MAGCHAR:
46 * Set in gl_flags if pattern contained a globbing character.
47 * GLOB_NOMAGIC:
48 * Same as GLOB_NOCHECK, but it will only append pattern if it did
49 * not contain any magic characters. [Used in csh style globbing]
50 * GLOB_ALTDIRFUNC:
51 * Use alternately specified directory access functions.
52 * GLOB_TILDE:
53 * expand ~user/foo to the /home/dir/of/user/foo
54 * GLOB_BRACE:
55 * expand {1,2}{a,b} to 1a 1b 2a 2b
56 * gl_matchc:
57 * Number of matches in the current invocation of glob.
58 * Copyright 1985, 1991 by Mortice Kern Systems Inc.  All rights reserved.
59 * Written by Eric Gisin.
60 */
61
62 #include <sys/param.h>
63 #include <sys/stat.h>
64 #pragma ident      "%Z%M% %I%      %E% SMI"
65
66 #include <ctype.h>
67 #include <dirent.h>
68 #include <errno.h>
69 #include <glob.h>
70 #include <limits.h>
71 #include <pwd.h>
72 #pragma weak _glob = glob
73 #pragma weak _globfree = globfree
74
75 #include "lint.h"
76 #include <stdio.h>
77 #include <unistd.h>
78 #include <limits.h>
79 #include <stdlib.h>
80 #include <string.h>
81 #include <unistd.h>
82 #include <wchar.h>
83 #include <wctype.h>
84 #include <dirent.h>
85 #include <sys/stat.h>
86 #include <glob.h>
87 #include <errno.h>
88 #include <fnmatch.h>
89
90 #define DOLLAR      '$'
91 #define DOT         '.'
92 #define EOS         '\0'
93 #define LBRACKET    '['

```

```

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

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

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

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

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

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

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

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

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

```

```

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

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

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

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

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

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

```

```

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

```

```

87     free((void *)gp->gl_pathv);
88
265     return (glob0(pattern, pglob, limitp, errfunc));
89     gp->gl_pathc = 0;
90     gp->gl_pathv = NULLCPP;
266 }
267
268 /*
269  * Recursive brace globbing helper. Tries to expand a single brace.
270  * If it succeeds then it invokes globexpl with the new pattern.
271  * If it fails then it tries to glob the rest of the pattern and returns.
272  * Do filename expansion.
273  */
274 static int
275 globexp2(const wcat_t *ptr, const wcat_t *pattern, glob_t *pglob,
276          struct glob_lim *limitp, int (*errfunc)(const char *, int))
277 {
278     int i, rv;
279     wcat_t *lm, *ls;
280     const wcat_t *pe, *pm, *pl;
281     wcat_t patbuf[MAXPATHLEN];
282     int rv;
283     size_t i;
284     size_t ipathc;
285     char *path;
286
287     /* copy part up to the brace */
288     for (lm = patbuf, pm = pattern; pm != ptr; *lm++ = *pm++)
289         ;
290     lm->w_at = 0;
291     lm->w_wc = EOS;
292     ls = lm;
293     if ((flags & GLOB_DOOFFS) == 0)
294         gp->gl_offs = 0;
295
296     /* Find the balanced brace */
297     for (i = 0, pe = ++ptr; pe->w_wc != EOS; pe++)
298         if (pe->w_wc == LBRACKET) {
299             /* Ignore everything between [] */
300             for (pm = pe++; pe->w_wc != RBRACKET &&
301                 pe->w_wc != EOS; pe++)
302                 ;
303             if (pe->w_wc == EOS) {
304                 /*
305                  * We could not find a matching RBRACKET.
306                  * Ignore and just look for RBRACE
307                  */
308                 pe = pm;
309             }
310             } else if (pe->w_wc == LBRACE) {
311                 i++;
312             } else if (pe->w_wc == RBRACE) {
313                 if (i == 0)
314                     break;
315                 i--;
316             }
317     if (!(flags & GLOB_APPEND)) {
318         gp->gl_pathc = 0;
319         gp->gl_pathn = gp->gl_offs + INITIAL;
320         gp->gl_pathv = (char **)malloc(sizeof(char *) * gp->gl_pathn);
321
322         /* Non matching braces; just glob the pattern */

```

```

313     if (i != 0 || pe->w_wc == EOS)
314         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;

316     for (i = 0, pl = pm = ptr; pm <= pe; pm++) {
317         switch (pm->w_wc) {
318             case LBRACKET:
319                 /* Ignore everything between [] */
320                 for (pl = pm++; pm->w_wc != RBRACKET && pm->w_wc != EOS;
321                     pm++)
322                     ;
323                 if (pm->w_wc == EOS) {
324                     /*
325                      * We could not find a matching RBRACKET.
326                      * Ignore and just look for RBRACE
327                      */
328                     pm = pl;
117                 for (i = 0; i < gp->gl_offs; ++i)
118                     gp->gl_pathv[i] = NULL;
329                 }
330                 break;

332             case LBRACE:
333                 i++;
334                 break;
121             if ((path = malloc(strlen(pattern)+1)) == NULL)
122                 return (GLOB_NOSPACE);

336             case RBRACE:
337                 if (i) {
338                     i--;
339                     break;
340                 }
341                 /* FALLTHROUGH */
342             case COMMA:
343                 if (i && pm->w_wc == COMMA)
344                     break;
345                 else {
346                     /* Append the current string */
347                     for (lm = ls; (pl < pm); *lm++ = *pl++)
348                         ;
124                     ipathc = gp->gl_pathc;
125                     rv = globit(0, pattern, gp, flags, errfn, &path);

127                 if (rv == GLOB_ABORTED) {
350                     /*
351                      * Append the rest of the pattern after the
352                      * 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.
353                     */
354                     for (pl = pe + 1;
355                         (*lm++ = *pl++).w_wc != EOS; /* */)
356                         ;

358                     /* Expand the current pattern */
359                     rv = globexpl(patbuf, pglob, limitp, errfunc);
360                     if (rv && rv != GLOB_NOMATCH)
361                         return (rv);

363                     /* move after the comma, to the next string */
364                     pl = pm + 1;
132                 free(path);
133                 return (GLOB_ABORTED);

```

```

365     }
366     break;

368     default:
369         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);
370     }
371     }
372     return (0);
373 }

377 /*
378  * expand tilde from the passwd file.
379  */
380 static const wcat_t *
381 globtilde(const wcat_t *pattern, wcat_t *patbuf, size_t patbuf_len,
382           glob_t *pglob)
383 {
384     struct passwd *pwd;
385     char *h;
386     const wcat_t *p;
387     wcat_t *b, *eb, *q;
388     size_t n, lenh;
389     wchar_t c;

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

394     /* Copy up to the end of the string or / */
395     eb = &patbuf[patbuf_len - 1];
396     for (p = pattern + 1, q = patbuf;
397         q < eb && p->w_wc != EOS && p->w_wc != SLASH; *q++ = *p++)
398         ;

400     q->w_at = 0;
401     q->w_wc = EOS;

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

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

```



```

423         return (pattern);
424     else
425         h = pwd->pw_dir;
426     }
427     gp->gl_pathp[gp->gl_pathc] = NULL;
428     free(path);
429
430     /* Copy the home directory */
431     lenh = strlen(h) + 1;
432     for (b = patbuf; b < eb && *h != EOS; b++) {
433         if ((n = mbtowc(&c, h, lenh)) > 0) {
434             h += n;
435             lenh -= n;
436             b->w_at = 0;
437             b->w_wc = c;
438         } else if (n < 0) {
439             return (pattern);
440         } else {
441             break;
442         }
443     }
444
445     /* Append the rest of the pattern */
446     while (b < eb && (*b++ = *p++)>.w_wc != EOS)
447         ;
448     b->w_at = 0;
449     b->w_wc = EOS;
450
451     return (patbuf);
452     return (rv);
453 }
454
455 static int
456 g_charclass(const wcat_t **patternp, wcat_t **bufnextp)
457 {
458     const wcat_t *pattern = *patternp + 1;
459     wcat_t *bufnext = *bufnextp;
460     const wcat_t *colon;
461     char cbuf[MB_LEN_MAX + 32];
462     wctype_t cc;
463     size_t len;
464
465     if ((colon = g_strchr(pattern, COLON)) == NULL ||
466         colon[1].w_wc != RBRACKET)
467         return (1); /* not a character class */
468
469     len = (size_t)(colon - pattern);
470     if (len + MB_LEN_MAX + 1 > sizeof (cbuf))
471         return (-1); /* invalid character class */
472
473     {
474         wchar_t w;
475         const wcat_t *s1 = pattern;
476         char *s2 = cbuf;
477         size_t n = len;
478
479         /* Copy the string. */
480         while (n > 0) {
481             w = (s1++)->w_wc;
482             /* Character class names must be ASCII. */
483             if (iswascii(w)) {
484                 n--;
485                 *s2++ = w;
486             } else {
487                 return (-1); /* invalid character class */
488             }
489         }
490     }

```

```

486         *s2 = EOS;
487     }
488     if ((cc = wctype(cbuf)) == 0)
489         return (-1); /* invalid character class */
490     bufnext->w_at = M_QUOTE;
491     (bufnext++)->w_wc = M_CLASS;
492     bufnext->w_at = 0;
493     (bufnext++)->w_wc = cc;
494     *bufnextp = bufnext;
495     *patternp += len + 3;
496
497     return (0);
498 }
499
500 /*
501 * The main glob() routine: compiles the pattern (optionally processing
502 * quotes), calls glob1() to do the real pattern matching, and finally
503 * sorts the list (unless unsorted operation is requested). Returns 0
504 * if things went well, nonzero if errors occurred. It is not an error
505 * to find no matches.
506 * Recursive routine to match glob pattern, and walk directories.
507 */
508 static int
509 glob0(const wcat_t *pattern, glob_t *pglob, struct glob_lim *limitp,
510       int (*errfunc)(const char *, int))
511 {
512     int
513     globit(size_t dend, const char *sp, glob_t *gp, int flags,
514           int (*errfn)(const char *, int), char **path)
515     {
516         const wcat_t *qpatnext;
517         int err, oldpathc;
518         wchar_t c;
519         int a;
520         wcat_t *bufnext, patbuf[MAXPATHLEN];
521         size_t n;
522         size_t m;
523         ssize_t end = 0; /* end of expanded directory */
524         char *pat = (char *)sp; /* pattern component */
525         char *dp = (*path) + dend;
526         int expand = 0; /* path has pattern */
527         char *cp;
528         struct stat64 sb;
529         DIR *dirp;
530         struct dirent64 *d;
531         int err;
532
533         qpatnext = globtilde(pattern, patbuf, MAXPATHLEN, pglob);
534         oldpathc = pglob->gl_pathc;
535         bufnext = patbuf;
536
537         /*
538          * We don't need to check for buffer overflow any more.
539          * The pattern has already been copied to an internal buffer.
540          */
541         while ((a = qpatnext->w_at), (c = (qpatnext++)->w_wc) != EOS) {
542             switch (c) {
543             case LBRACKET:
544                 if (a != 0) {
545                     bufnext->w_at = a;
546                     (bufnext++)->w_wc = c;
547                     break;
548                 }
549                 for (;;)
550                     switch (*dp++ = *(unsigned char *)sp++) {
551                     case '\0': /* end of source path */
552                         if (expand)
553                             goto Expand;

```

```

178     else {
179         if (!(flags & GLOB_NOCHECK) ||
180             flags & (GLOB_NOCHECK|GLOB_MARK))
181             if (stat64(*path, &sb) < 0) {
182                 return (0);
183             }
184         a = qpatnext->w_at;
185         c = qpatnext->w_wc;
186         if (a == 0 && c == NOT)
187             ++qpatnext;
188         if (qpatnext->w_wc == EOS ||
189             g_strchr(qpatnext+1, RBRACKET) == NULL) {
190             bufnext->w_at = 0;
191             (bufnext++)->w_wc = LBRACKET;
192             if (a == 0 && c == NOT)
193                 --qpatnext;
194             break;
195             if (flags & GLOB_MARK && S_ISDIR(sb.st_mode)) {
196                 *dp = '\0';
197                 *--dp = '/';
198             }
199             bufnext->w_at = M_QUOTE;
200             (bufnext++)->w_wc = M_SET;
201             if (a == 0 && c == NOT) {
202                 bufnext->w_at = M_QUOTE;
203                 (bufnext++)->w_wc = M_NOT;
204             }
205             a = qpatnext->w_at;
206             c = (qpatnext++)->w_wc;
207             do {
208                 if (a == 0 && c == LBRACKET &&
209                     qpatnext->w_wc == COLON) {
210                     do {
211                         err = g_charclass(&qpatnext,
212                                             &bufnext);
213                         if (err)
214                             break;
215                         a = qpatnext->w_at;
216                         c = (qpatnext++)->w_wc;
217                     } while (a == 0 && c == LBRACKET &&
218                             qpatnext->w_wc == COLON);
219                     if (err == -1 &&
220                         !(pglob->gl_flags & GLOB_NOCHECK))
221                         return (GLOB_NOMATCH);
222                     if (a == 0 && c == RBRACKET)
223                         break;
224                 }
225                 bufnext->w_at = a;
226                 (bufnext++)->w_wc = c;
227                 if (qpatnext->w_at == 0 &&
228                     qpatnext->w_wc == RANGE) {
229                     a = qpatnext[1].w_at;
230                     c = qpatnext[1].w_wc;
231                     if (qpatnext[1].w_at != 0 ||
232                         qpatnext[1].w_wc != RBRACKET) {
233                         bufnext->w_at = M_QUOTE;
234                         (bufnext++)->w_wc = M_RNG;
235                         bufnext->w_at = a;
236                         (bufnext++)->w_wc = c;
237                         qpatnext += 2;
238                     }
239                 }
240             } while (a != 0 || c != RBRACKET);
241             pglob->gl_flags |= GLOB_MAGCHAR;

```

```

590             bufnext->w_at = M_QUOTE;
591             (bufnext++)->w_wc = M_END;
592             break;
593         case QUESTION:
594             if (a != 0) {
595                 bufnext->w_at = a;
596                 (bufnext++)->w_wc = c;
597                 break;
598             }
599             pglob->gl_flags |= GLOB_MAGCHAR;
600             bufnext->w_at = M_QUOTE;
601             (bufnext++)->w_wc = M_ONE;
602             break;
603         case STAR:
604             if (a != 0) {
605                 bufnext->w_at = a;
606                 (bufnext++)->w_wc = c;
607                 break;
608             }
609             pglob->gl_flags |= GLOB_MAGCHAR;
610             /*
611              * collapse adjacent stars to one,
612              * to avoid exponential behavior
613              */
614             if (bufnext == patbuf ||
615                 bufnext[-1].w_at != M_QUOTE ||
616                 bufnext[-1].w_wc != M_ALL) {
617                 bufnext->w_at = M_QUOTE;
618                 (bufnext++)->w_wc = M_ALL;
619             }
620             break;
621         default:
622             bufnext->w_at = a;
623             (bufnext++)->w_wc = c;
624             break;
625     }
626 }
627 bufnext->w_at = 0;
628 bufnext->w_wc = EOS;
629 #ifdef DEBUG
630     qprintf("glob0:glob1:patbuf", patbuf);
631 #endif
632
633     if ((err = glob1(patbuf, patbuf+MAXPATHLEN-1, pglob, limitp, errfunc))
634         != 0)
635         return (err);
636
637     /*
638      * If there was no match we are going to append the pattern
639      * if GLOB_NOCHECK was specified or if GLOB_NOMAGIC was specified
640      * and the pattern did not contain any magic characters
641      * GLOB_NOMAGIC is there just for compatibility with csh.
642      */
643     if (pglob->gl_pathc == oldpathc) {
644         if ((pglob->gl_flags & GLOB_NOCHECK) ||
645             ((pglob->gl_flags & GLOB_NOMAGIC) &&
646              !(pglob->gl_flags & GLOB_MAGCHAR)))
647             return (globextend(pattern, pglob, limitp, NULL));
648         else
649             return (GLOB_NOMATCH);
650     }
651     if (!(pglob->gl_flags & GLOB_NOSORT)) {
652         if ((pglob->gl_flags & GLOB_KEEPCONCAT)) {
653             /* Keep the paths and stat info synced during sort */
654             struct glob_path_stat *path_stat;
655             int i;

```

```

656     int n = pglob->gl_pathc - oldpathc;
657     int o = pglob->gl_offs + oldpathc;

659     if ((path_stat = calloc(n, sizeof (*path_stat))) ==
660         NULL)
188         if (append(gp, *path) < 0) {
661             return (GLOB_NOSPACE);
662         }
663         for (i = 0; i < n; i++) {
664             path_stat[i].gps_path = pglob->gl_pathv[o + i];
665             path_stat[i].gps_stat = pglob->gl_statv[o + i];
666         }
667         qsort(path_stat, n, sizeof (*path_stat), compare_gps);
668         for (i = 0; i < n; i++) {
669             pglob->gl_pathv[o + i] = path_stat[i].gps_path;
670             pglob->gl_statv[o + i] = path_stat[i].gps_stat;
671         }
672     } else {
673         free(path_stat);
674         qsort(pglob->gl_pathv + pglob->gl_offs + oldpathc,
675             pglob->gl_pathc - oldpathc, sizeof (char *),
676             compare);
677     }
678     return (0);
679 }
192 }
193 /*NOTREACHED*/

681 static int
682 compare(const void *p, const void *q)
683 {
684     return (strcmp(*(char **)p, *(char **)q));
685 }
195     case '*':
196     case '?':
197     case '[':
198     case '\\':
199         ++expand;
200         break;

687 static int
688 compare_gps(const void *_p, const void *_q)
689 {
690     const struct glob_path_stat *p = (const struct glob_path_stat *)_p;
691     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;

693     return (strcmp(p->gps_path, q->gps_path));
694 }

696 static int
697 glob1(wcat_t *pattern, wcat_t *pattern_last, glob_t *pglob,
698     struct glob_lim *limitp, int (*errfunc)(const char *, int))
699 {
700     wcat_t pathbuf[MAXPATHLEN];

702     /* A null pathname is invalid -- POSIX 1003.1 sect. 2.4. */
703     if (pattern->w_wc == EOS)
704         return (0);
705     return (glob2(pathbuf, pathbuf+MAXPATHLEN-1,
706         pathbuf, pathbuf+MAXPATHLEN-1,

```

```

707     pattern, pattern_last, pglob, limitp, errfunc));
708 }

710 /*
711 * The functions glob2 and glob3 are mutually recursive; there is one level
712 * of recursion for each segment in the pattern that contains one or more
713 * meta characters.
714 */
715 static int
716 glob2(wcat_t *pathbuf, wcat_t *pathbuf_last, wcat_t *pathend,
717     wcat_t *pathend_last, wcat_t *pattern, wcat_t *pattern_last,
718     glob_t *pglob, struct glob_lim *limitp, int (*errfunc)(const char *, int))
719 {
720     struct stat sb;
721     wcat_t *p, *q;
722     int anymeta;

724     /*
725     * Loop over pattern segments until end of pattern or until
726     * segment with meta character found.
727     */
728     for (anymeta = 0; ; ) {
729         if (pattern->w_wc == EOS) { /* End of pattern? */
730             pathend->w_at = 0;
731             pathend->w_wc = EOS;

733             if ((pglob->gl_flags & GLOB_LIMIT) &&
734                 limitp->glim_stat++ >= GLOB_LIMIT_STAT) {
735                 errno = 0;
736                 pathend->w_at = 0;
737                 (pathend++)->w_wc = SEP;
738                 pathend->w_at = 0;
739                 pathend->w_wc = EOS;
740                 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 (errno != 0 && errfn(*path, errno) != 0 ||
215                     flags&GLOB_ERR) {
216                     return (GLOB_ABORTED);
741             }
742             if (g_lstat(pathbuf, &sb, pglob))
743                 return (0);

745             if (((pglob->gl_flags & GLOB_MARK) &&
746                 (pathend[-1].w_at != 0 ||
747                 pathend[-1].w_wc != SEP)) &&
748                 (S_ISDIR(sb.st_mode) ||
749                 (S_ISLNK(sb.st_mode) &&
750                 (g_stat(pathbuf, &sb, pglob) == 0) &&
751                 S_ISDIR(sb.st_mode)))) {
752                 if (pathend+1 > pathend_last)
753                     return (GLOB_NOSPACE);
754                 pathend->w_at = 0;
755                 (pathend++)->w_wc = SEP;
756                 pathend->w_at = 0;
757                 pathend->w_wc = EOS;
758             }
759             ++pglob->gl_matchc;
760             return (globextend(pathbuf, pglob, limitp, &sb));
761         }

763     /* Find end of next segment, copy tentatively to pathend. */
764     q = pathend;

```

```

765     p = pattern;
766     while (p->w_wc != EOS && p->w_wc != SEP) {
767         if (ismeta(*p))
768             anymeta = 1;
769         if (q+1 > pathend_last)
770             /* extract pattern component */
771             n = sp - pat;
772         if ((cp = malloc(n)) == NULL) {
773             (void) closedir(dirp);
774             return (GLOB_NOSPACE);
775         }
776         *q++ = *p++;
777     }
778     pat = memcpy(cp, pat, n);
779     pat[n-1] = '\\0';
780     if (*--sp != '\\0')
781         flags |= GLOB_CHECK;
782
783     if (!anymeta) { /* No expansion, do next segment. */
784         pathend = q;
785         pattern = p;
786         while (pattern->w_wc == SEP) {
787             if (pathend+1 > pathend_last)
788                 /* expand path to max. expansion */
789                 n = dp - *path;
790             *path = realloc(*path,
791                 strlen(*path) + NAME_MAX + strlen(sp) + 1);
792             if (*path == NULL) {
793                 (void) closedir(dirp);
794                 free(pat);
795                 return (GLOB_NOSPACE);
796             }
797             *pathend++ = *pattern++;
798         }
799     } else {
800         /* Need expansion, recurse. */
801         return (glob3(pathbuf, pathbuf_last, pathend,
802             pathend_last, pattern, p, pattern_last,
803             pglob, limitp, errfunc));
804     }
805 }
806 /* NOTREACHED */
807
808 dp = (*path) + n;
809
810 static int
811 glob3(wcat_t *pathbuf, wcat_t *pathbuf_last, wcat_t *pathend,
812 wcat_t *pathend_last, wcat_t *pattern, wcat_t *restpattern,
813 wcat_t *restpattern_last, glob_t *pglob, struct glob_lim *limitp,
814 int (*errfunc)(const char *, int))
815 {
816     struct dirent *dp;
817     DIR *dirp;
818     int err;
819     char buf[MAXPATHLEN];
820
821     /*
822      * The readdirfunc declaration can't be prototyped, because it is
823      * assigned, below, to two functions which are prototyped in glob.h
824      * and dirent.h as taking pointers to differently typed opaque
825      * structures.
826      */
827     struct dirent *(*readdirfunc)(void *);
828
829     if (pathend > pathend_last)
830         return (GLOB_NOSPACE);
831     pathend->w_at = 0;
832     pathend->w_wc = EOS;

```

```

815     errno = 0;
816
817     if ((dirp = g_opendir(pathbuf, pglob)) == NULL) {
818         /* TODO: don't call for ENOENT or ENOTDIR? */
819         if (errfunc) {
820             if (g_Ctoc(pathbuf, buf, sizeof(buf)))
821                 return (GLOB_ABORTED);
822             if (errfunc(buf, errno) ||
823                 pglob->gl_flags & GLOB_ERR)
824                 return (GLOB_ABORTED);
825         }
826         return (0);
827     }
828
829     /* read directory and match entries */
830     err = 0;
831
832     /* Search directory for matching names. */
833     if (pglob->gl_flags & GLOB_ALTDIRFUNC)
834         readdirfunc = pglob->gl_readdir;
835     else
836         readdirfunc = (struct dirent *(*)(void *))readdir;
837     while ((dp = (*readdirfunc)(dirp)) != NULL) {
838         char *sc;
839         wcat_t *dc;
840         size_t n, lensc;
841         wchar_t w;
842
843         if ((pglob->gl_flags & GLOB_LIMIT) &&
844             limitp->glim_readdir++ >= GLOB_LIMIT_READDIR) {
845             errno = 0;
846             pathend->w_at = 0;
847             (pathend++)->w_wc = SEP;
848             pathend->w_at = 0;
849             pathend->w_wc = EOS;
850             err = GLOB_NOSPACE;
851             break;
852         }
853
854         /* Initial DOT must be matched literally. */
855         if (dp->d_name[0] == DOT && pattern->w_wc != DOT)
856             while ((d = readdir64(dirp)) != NULL) {
857                 cp = d->d_name;
858                 if ((flags & GLOB_NOESCAPE)
859                     ? fnmatch(pat, cp, FNM_PERIOD|FNM_NOESCAPE)
860                     : fnmatch(pat, cp, FNM_PERIOD))
861                     continue;
862                 dc = pathend;
863                 sc = dp->d_name;
864                 lensc = strlen(sc) + 1;
865                 while (dc < pathend_last) {
866                     if ((n = mbtowl(&w, sc, lensc)) <= 0) {
867                         sc += 1;
868                         lensc -= 1;
869                         dc->w_at = 0;
870                         dc->w_wc = EOS;
871                     } else {
872                         sc += n;
873                         lensc -= n;
874                         dc->w_at = 0;
875                         dc->w_wc = w;
876                     }
877                 }
878                 dc++;
879                 if (n <= 0)
880                     break;
881             }

```

```

875     if (dc >= pathend_last) {
876         dc->w_at = 0;
877         dc->w_wc = EOS;
878         err = GLOB_NOSPACE;
879         break;
880     }
881     if (n < 0) {
882         err = GLOB_NOMATCH;
883         break;
884     }
885
886     if (!match(pathend, pattern, restpattern, GLOB_LIMIT_RECUR)) {
887         pathend->w_at = 0;
888         pathend->w_wc = EOS;
889         continue;
890     }
891     err = glob2(pathbuf, pathbuf_last, --dc, pathend_last,
892               restpattern, restpattern_last, pglob, limitp,
893               errfunc);
894     if (err)
895         break;
896
897     n = strlen(cp);
898     (void) memcpy((*path) + end, cp, n);
899     m = dp - *path;
900     err = globit(end+n, sp, gp, flags, errfn, path);
901     dp = (*path) + m; /* globit can move path */
902     if (err != 0)
903         break;
904
905     if (pglob->gl_flags & GLOB_ALTDIRFUNC)
906         (*pglob->gl_closedir)(dirp);
907     else
908         closedir(dirp);
909     (void) closedir(dirp);
910     free(pat);
911     return (err);
912 }
913 /* NOTREACHED */
914 }

```

```

906 /*
907 * Extend the gl_pathv member of a glob_t structure to accommodate a new item,
908 * add the new item, and update gl_pathc.
909 *
910 * This assumes the BSD realloc, which only copies the block when its size
911 * crosses a power-of-two boundary; for v7 realloc, this would cause quadratic
912 * behavior.
913 *
914 * Return 0 if new item added, error code if memory couldn't be allocated.
915 *
916 * Invariant of the glob_t structure:
917 *   Either gl_pathc is zero and gl_pathv is NULL; or gl_pathc > 0 and
918 *   gl_pathv points to (gl_offs + gl_pathc + 1) items.
919 * Comparison routine for two name arguments, called by qsort.
920 */
921 static int
922 globextend(const wcat_t *path, glob_t *pglob, struct glob_lim *limitp,
923            struct stat *sb)
924 {
925     int i;
926     pstricmp(const void *npp1, const void *npp2)
927     {
928         char **pathv;
929         ssize_t i;
930         size_t newn, len;
931         char *copy = NULL;

```

```

928     const wcat_t *p;
929     struct stat **statv;
930     char junk[MB_LEN_MAX];
931     int n;
932
933     newn = 2 + pglob->gl_pathc + pglob->gl_offs;
934     if (pglob->gl_offs >= INT_MAX ||
935         pglob->gl_pathc >= INT_MAX ||
936         newn >= INT_MAX ||
937         SIZE_MAX / sizeof (*pathv) <= newn ||
938         SIZE_MAX / sizeof (*statv) <= newn) {
939         nospace:
940         for (i = pglob->gl_offs; i < (ssize_t)(newn - 2); i++) {
941             if (pglob->gl_pathv && pglob->gl_pathv[i])
942                 free(pglob->gl_pathv[i]);
943             if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
944                 pglob->gl_statv && pglob->gl_statv[i])
945                 free(pglob->gl_statv[i]);
946         }
947         if (pglob->gl_pathv) {
948             free(pglob->gl_pathv);
949             pglob->gl_pathv = NULL;
950         }
951         if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
952             pglob->gl_statv) {
953             free(pglob->gl_statv);
954             pglob->gl_statv = NULL;
955         }
956         return (GLOB_NOSPACE);
957     }
958
959     pathv = realloc(pglob->gl_pathv, newn * sizeof (*pathv));
960     if (pathv == NULL)
961         goto nospace;
962     if (pglob->gl_pathv == NULL && pglob->gl_offs > 0) {
963         /* first time around -- clear initial gl_offs items */
964         pathv += pglob->gl_offs;
965         for (i = pglob->gl_offs; --i >= 0; )
966             *--pathv = NULL;
967     }
968     pglob->gl_pathv = pathv;
969
970     if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0) {
971         statv = realloc(pglob->gl_statv, newn * sizeof (*statv));
972         if (statv == NULL)
973             goto nospace;
974         if (pglob->gl_statv == NULL && pglob->gl_offs > 0) {
975             /* first time around -- clear initial gl_offs items */
976             statv += pglob->gl_offs;
977             for (i = pglob->gl_offs; --i >= 0; )
978                 *--statv = NULL;
979         }
980         pglob->gl_statv = statv;
981         if (sb == NULL)
982             statv[pglob->gl_offs + pglob->gl_pathc] = NULL;
983     }
984     else {
985         limitp->glim_malloc += sizeof (**statv);
986         if ((pglob->gl_flags & GLOB_LIMIT) &&
987             limitp->glim_malloc >= GLOB_LIMIT_MALLOC) {
988             errno = 0;
989             return (GLOB_NOSPACE);
990         }
991         if ((statv[pglob->gl_offs + pglob->gl_pathc] =
992             malloc(sizeof (**statv))) == NULL)
993             goto copy_error;
994         memcpy(statv[pglob->gl_offs + pglob->gl_pathc], sb,

```

```

994         sizeof (*sb));
995     }
996     statv[pglob->gl_offs + pglob->gl_pathc + 1] = NULL;
997 }

999     len = MB_LEN_MAX;
1000     p = path;
1001     while ((n = wctomb(junk, p->w_wc) > 0) {
1002         len += n;
1003         if ((p++)->w_wc == EOS)
1004             break;
1005     }
1006     if (n < 0)
1007         return (GLOB_NOMATCH);

1009     limitp->glim_malloc += len;
1010     if ((copy = malloc(len)) != NULL) {
1011         if (g_Ctoc(path, copy, len)) {
1012             free(copy);
1013             return (GLOB_NOSPACE);
1014         }
1015         pathv[pglob->gl_offs + pglob->gl_pathc++] = copy;
1016     }
1017     pathv[pglob->gl_offs + pglob->gl_pathc] = NULL;

1019     if ((pglob->gl_flags & GLOB_LIMIT) &&
1020         (newn * sizeof (*pathv) + limitp->glim_malloc >
1021          GLOB_LIMIT_MALLOC) {
1022         errno = 0;
1023         return (GLOB_NOSPACE);
1024     }
1025     copy_error:
1026     return (copy == NULL ? GLOB_NOSPACE : 0);
1027     return (strcoll(*(char **)npp1, *(char **)npp2));
}

1030 /*
1031  * pattern matching function for filenames.  Each occurrence of the *
1032  * pattern causes a recursion level.
1033  * Add a new matched filename to the glob_t structure, increasing the
1034  * size of that array, as required.
1035  */
1036 static int
1037 match(wcat_t *name, wcat_t *pat, wcat_t *patend, int recur)
1038 {
1039     int
1040     append(glob_t *gp, const char *str)
1041     {
1042         int ok, negate_range;
1043         wcat_t c, k;
1044         char *cp;

1045         if (recur-- == 0)
1046             return (1);
1047         if ((cp = malloc(strlen(str)+1)) == NULL)
1048             return (GLOB_NOSPACE);
1049         gp->gl_pathp[gp->gl_pathc++] = strcpy(cp, str);

1050         while (pat < patend) {
1051             c = *pat++;
1052             switch (c.w_wc) {
1053             case M_ALL:
1054                 if (c.w_at != M_QUOTE) {
1055                     k = *name++;
1056                     if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1057                         return (0);
1058                 }
1059             }

```

```

1051         break;
1052         if ((gp->gl_pathc + gp->gl_offs) >= gp->gl_pathn) {
1053             gp->gl_pathn *= 2;
1054             gp->gl_pathv = (char **)realloc((void *)gp->gl_pathv,
1055             gp->gl_pathn * sizeof (char *));
1056             if (gp->gl_pathv == NULLCPP)
1057                 return (GLOB_NOSPACE);
1058             gp->gl_pathp = gp->gl_pathv + gp->gl_offs;
1059         }
1060         while (pat < patend && pat->w_at == M_QUOTE &&
1061                pat->w_wc == M_ALL)
1062             pat++; /* eat consecutive '*' */
1063         if (pat == patend)
1064             return (1);
1065         do {
1066             if (match(name, pat, patend, recur))
1067                 return (1);
1068             } while ((name++)->w_wc != EOS);
1069         return (0);

1070     case M_ONE:
1071         if (c.w_at != M_QUOTE) {
1072             k = *name++;
1073             if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1074                 return (0);
1075             break;
1076         }
1077         if ((name++)->w_wc == EOS)
1078             return (0);
1079         break;

1080     case M_SET:
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         ok = 0;
1088         if ((k = *name+).w_wc == EOS)
1089             return (0);
1090         if ((negate_range = (pat->w_at == M_QUOTE &&
1091                pat->w_wc == M_NOT)) != 0)
1092             ++pat;
1093         while (((c = *pat+).w_at != M_QUOTE) ||
1094                c.w_wc != M_END) {
1095             if (c.w_at == M_QUOTE && c.w_wc == M_CLASS) {
1096                 wcat_t cc;

1097                 cc.w_at = pat->w_at;
1098                 cc.w_wc = pat->w_wc;
1099                 if (iswctype(k.w_wc, cc.w_wc))
1100                     ok = 1;
1101                 ++pat;
1102             }
1103             if (pat->w_at == M_QUOTE &&
1104                 pat->w_wc == M_RNG) {
1105                 if (c.w_wc <= k.w_wc &&
1106                     k.w_wc <= pat[1].w_wc)
1107                     ok = 1;
1108                 pat += 2;
1109             } else if (c.w_wc == k.w_wc)
1110                 ok = 1;
1111         }
1112         if (ok == negate_range)
1113             return (0);
1114         break;

1115     default:

```

```

1110         k = *name++;
1111         if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1112             return (0);
1113         break;
1114     }
1115 }
1116 return (name->w_wc == EOS);
1117 }

```

```

1119 /* Free allocated data belonging to a glob_t structure. */

```

```

1120 void
1121 globfree(glob_t *pglob)
1122 {
1123     int i;
1124     char **pp;
1125
1126     if (pglob->gl_pathv != NULL) {
1127         pp = pglob->gl_pathv + pglob->gl_offs;
1128         for (i = pglob->gl_pathc; i--; ++pp)
1129             if (*pp)
1130                 free(*pp);
1131         free(pglob->gl_pathv);
1132         pglob->gl_pathv = NULL;
1133     }
1134     if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
1135         pglob->gl_statv != NULL) {
1136         for (i = 0; i < pglob->gl_pathc; i++) {
1137             if (pglob->gl_statv[i] != NULL)
1138                 free(pglob->gl_statv[i]);
1139         }
1140         free(pglob->gl_statv);
1141         pglob->gl_statv = NULL;
1142     }
1143 }

```

```

1145 static DIR *
1146 g_opendir(wcat_t *str, glob_t *pglob)

```

```

1147 {
1148     char buf[MAXPATHLEN];
1149
1150     if (str->w_wc == EOS)
1151         strcpy(buf, ".", sizeof (buf));
1152     else {
1153         if (g_Ctoc(str, buf, sizeof (buf)))
1154             return (NULL);
1155     }

```

```

1157     if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1158         return ((*pglob->gl_opendir)(buf));

```

```

1160     return (opendir(buf));
1161 }

```

```

1163 static int
1164 g_lstat(wcat_t *fn, struct stat *sb, glob_t *pglob)

```

```

1165 {
1166     char buf[MAXPATHLEN];
1167
1168     if (g_Ctoc(fn, buf, sizeof (buf)))
1169         return (-1);
1170     if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1171         return ((*pglob->gl_lstat)(buf, sb));
1172     return (lstat(buf, sb));
1173 }

```

```

1175 static int

```

```

1176 g_stat(wcat_t *fn, struct stat *sb, glob_t *pglob)
1177 {
1178     char buf[MAXPATHLEN];

```

```

1180     if (g_Ctoc(fn, buf, sizeof (buf)))
1181         return (-1);
1182     if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1183         return ((*pglob->gl_stat)(buf, sb));
1184     return (stat(buf, sb));
1185 }

```

```

1187 static wcat_t *

```

```

1188 g_strchr(const wcat_t *str, wchar_t ch)
1189 {
1190     do {
1191         if (str->w_at == 0 && str->w_wc == ch)
1192             return ((wcat_t *)str);
1193     } while ((str++)->w_wc != EOS);
1194     return (NULL);
1195 }

```

```

1197 static int

```

```

1198 g_Ctoc(const wcat_t *str, char *buf, uint_t len)
1199 {
1200     int n;
1201     wchar_t w;
1202
1203     while (len >= MB_LEN_MAX) {
1204         w = (str++)->w_wc;
1205         if ((n = wctomb(buf, w)) > 0) {
1206             len -= n;
1207             buf += n;
1208         }
1209         if (n < 0)
1210             break;
1211         if (w == EOS)
1212             return (0);
1213     }
1214     return (1);
1215 }

```

```

1217 #ifndef DEBUG

```

```

1218 static void
1219 qprintf(const char *str, wcat_t *s)
1220 {
1221     wcat_t *p;
1222
1223     (void) printf("%s:\n", str);
1224     for (p = s; p->w_wc != EOS; p++)
1225         (void) printf("%wc", p->w_wc);
1226     (void) printf("\n");
1227     for (p = s; p->w_wc != EOS; p++)
1228         (void) printf("%c", p->w_at & M_PROTECT ? "' ' : ' ');
1229     (void) printf("\n");
1230     for (p = s; p->w_wc != EOS; p++)
1231         (void) printf("%c", ismeta(*p) ? '_ ' : ' ');
1232     (void) printf("\n");
1233 }
1234 #endif

```

```

*****
17719 Wed Jan 30 08:38:42 2013
new/usr/src/man/man3c/glob.3c
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
*****
1 \" te
2.\" Copyright (c) 1992, X/Open Company Limited. All Rights Reserved.
3.\" Portions Copyright (c) 2003, Sun Microsystems, Inc. All Rights Reserved.
4.\" Portions Copyright (c) 2012, Gary Mills
5.\" Copyright (c) 1992, X/Open Company Limited. All Rights Reserved. Portions C
6.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
7.\" http://www.opengroup.org/bookstore/.
8.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
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>
51.
52.\"#Bint \fR \fBglob\fR(\fBconst char *restrict\fR \fIpattern\fR, \fBint\fR \fIflg
53.\" \fBint\fR(\fR\fIerrfunc\fR)(const char *\fIepath\fR, int \fIeerrno)\fR,
54.\" \fBglob_t *restrict\fR \fIpglob\fR);
55.\".fi
56.
57.\"LP
58.\"nf
59.\"#Bvoid \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 */
96 size_t  gl_pathc;    /* count of paths matched by */
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. */
101 char    **gl_pathv; /* pointer to list of matched */
102         /* path names */
103         /* slots to reserve at beginning */
104         /* of gl_pathv */
105 .fi
106 .LP
107 The \fBglob()\fR function stores the number of matched path names into
108 \fIpglob\>\fR and a pointer to a list of pointers to path
109 names into \fIpglob\>\fR. The path names are in sort order as
110 defined by the current setting of the \fBLC_COLLATE\fR category. The first
111 pointer after the last path name is a \fBNULL\fR pointer. If the pattern does
112 not match any path names, the returned number of matched paths is set to 0, and
113 the contents of \fIpglob\>\fR are implementation-dependent.
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 \fIgl_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 \fBuffsrestore\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

```

```

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

260 .sp
261 .ne 2
262 .na
263 \fB\fBGLOB_QUOTE\fR\fR
264 .ad
265 .RS 17n
266 This option has no effect and is included for backwards
267 compatibility with older sources.
268 .RE

270 .sp
271 .ne 2
272 .na
273 \fB\fBGLOB_TILDE\fR\fR
274 .ad
275 .RS 17n
276 Expand patterns that start with '~' to user name home
277 directories.
278 .RE

280 .sp
281 .ne 2
282 .na
283 \fB\fBGLOB_LIMIT\fR\fR
284 .ad
285 .RS 17n
286 Limit the amount of memory used by matches to \fIARG_MAX\fR.
287 This option should be set for programs that can be coerced
288 to a denial of service attack via patterns that
289 expand to a very large number of matches, such as a long
290 string of '*/*/*/*/*'.
291 .RE

293 .sp
294 .ne 2
295 .na
296 \fB\fBGLOB_KEEPSTAT\fR\fR
297 .ad
298 .RS 17n
299 Retain a copy of the \fBstat\fR(2) information retrieved for
300 matching paths in the \fIgl_statv\fR array:
301 .sp
302 .nf
303 struct stat **gl_statv;
304 .fi
305 .sp
306 This option may be used to avoid \fBlstat\fR(2) lookups in
307 cases where they are expensive.
308 .RE

310 .sp
311 .LP
312 The \fBGLOB_APPEND\fR flag can be used to append a new set of path names to
313 those found in a previous call to \fBglob()\fR. The following rules apply when
314 two or more calls to \fBglob()\fR are made with the same value of \fIpglob\fR
315 and without intervening calls to \fBglobfree()\fR:
316 .RS +4
317 .TP
318 1.

```

```

319 The first such call must not set \fBGLOB_APPEND\fR. All subsequent calls
320 must set it.
321 .RE
322 .RS +4
323 .TP
324 2.
325 All the calls must set \fBGLOB_DOOFFS\fR or all must not set it.
326 .RE
327 .RS +4
328 .TP
329 3.
330 After the second call, \fIpglob\<mi>\fR\fBgl_pathv\fR points to a list
331 containing the following:
332 .RS +4
333 .TP
334 a.
335 Zero or more \fINULL\fR pointers, as specified by \fBGLOB_DOOFFS\fR and
336 \fIpglob\<mi>\fR\fBgl_offs\fR.
337 .RE
338 .RS +4
339 .TP
340 b.
341 Pointers to the path names that were in the \fIpglob\<mi>\fR\fBgl_pathv\fR
342 list before the call, in the same order as before.
343 .RE
344 .RS +4
345 .TP
346 c.
347 Pointers to the new path names generated by the second call, in the
348 specified order.
349 .RE
350 .RE
351 .RS +4
352 .TP
353 4.
354 The count returned in \fIpglob\<mi>\fR\fBgl_pathc\fR will be the total
355 number of path names from the two calls.
356 .RE
357 .RS +4
358 .TP
359 5.
360 The application can change any of the fields after a call to \fBglob()\fR.
361 If it does, it must reset them to the original value before a subsequent call,
362 using the same \fIpglob\fR value, to \fBglobfree()\fR or \fBglob()\fR with the
363 \fBGLOB_APPEND\fR flag.
364 .RE
365 .SS "\fIerrfunc\fR and \fIepath\fR Arguments"
366 .sp
367 .LP
368 If, during the search, a directory is encountered that cannot be opened or read
369 and \fIerrfunc\fR is not a \fINULL\fR pointer, \fBglob()\fR calls
370 \fB\<mi>\fR\fI*errfunc\<mi>\fR with two arguments:
371 .RS +4
372 .TP
373 1.
374 The \fIepath\fR argument is a pointer to the path that failed.
375 .RE
376 .RS +4
377 .TP
378 2.
379 The \fIeerrno\fR argument is the value of \fIerrno\fR from the failure, as
380 set by the \fBopendir\fR(3C), \fBreaddir\fR(3C) or \fBstat\fR(2) functions.
381 (Other values may be used to report other errors not explicitly documented for
382 those functions.)
383 .RE

```

```

385 .sp
386 .LP
387 If \fb(\fr\fi*errfunc\fr\fb)\fr is called and returns non-zero, or if the
388 \fbGLOB_ERR\fr flag is set in \fiflags\fr, \fbglob()\fr stops the scan and
389 returns \fbGLOB_ABORTED\fr after setting \figl_pathc\fr and \figl_pathv\fr in
390 \figglob\fr to reflect the paths already scanned. If \fbGLOB_ERR\fr is not set
391 and either \fierrfunc\fr is a \fINULL\fr pointer or
392 \fb(\fr\fi*errfunc\fr\fb)\fr returns 0, the error is ignored.
393 .SH RETURN VALUES
394 The following constants are defined as error return values for \fbglob()\fr:
395 .sp
396 .LP
397 On successful completion, \fbglob()\fr returns zero.
398 In addition the fields of pglob contain the values described below:
399 .sp
400 .ne 2
401 .na
402 \fb\fbgl_pathc\fr\fr
403 \fb\fbGLOB_ABORTED\fr\fr
404 .ad
405 .RS 16n
406 Contains the total number of matched pathnames so far.
407 This includes other matches from previous invocations of
408 \fbglob()\fr if \fbGLOB_APPEND\fr was specified.
409 The scan was stopped because \fbGLOB_ERR\fr was set or
410 \fb(\fr\fi*errfunc\fr\fb)\fr returned non-zero.
411 .RE
412 .sp
413 .ne 2
414 .na
415 \fb\fbgl_matchc\fr\fr
416 \fb\fbGLOB_NOMATCH\fr\fr
417 .ad
418 .RS 16n
419 Contains the number of matched pathnames in the current
420 invocation of \fbglob()\fr.
421 The pattern does not match any existing path name, and \fbGLOB_NOCHECK\fr was
422 not set in flags.
423 .RE
424 .sp
425 .ne 2
426 .na
427 \fb\fbgl_flags\fr\fr
428 \fb\fbGLOG_NOSPACE\fr\fr
429 .ad
430 .RS 16n
431 Contains a copy of the flags parameter with the bit
432 \fbGLOB_MAGCHAR\fr set if pattern contained any of the special
433 characters '*', '?', or '[', cleared if not.
434 An attempt to allocate memory failed.
435 .RE
436 .sp
437 .ne 2
438 .na
439 \fb\fbgl_pathv\fr\fr
440 .ad
441 .RS 16n
442 Contains a pointer to a null-terminated list of matched
443 pathnames. However, if \fbgl_pathc\fr is zero, the contents of
444 \fbgl_pathv\fr are undefined.
445 .RE

```

```

273 .LP
274 If \fb(\fr\fi*errfunc\fr\fb)\fr is called and returns non-zero, or if the
275 \fbGLOB_ERR\fr flag is set in \fiflags\fr, \fbglob()\fr stops the scan and
276 returns \fbGLOB_ABORTED\fr after setting \figl_pathc\fr and \figl_pathv\fr in
277 \figglob\fr to reflect the paths already scanned. If \fbGLOB_ERR\fr is not set
278 and either \fierrfunc\fr is a \fINULL\fr pointer or
279 \fb(\fr\fi*errfunc\fr\fb)\fr returns 0, the error is ignored.
280 .SH RETURN VALUES
281 .sp
282 .ne 2
283 .na
284 \fb\fbgl_statv\fr\fr
285 .ad
286 .RS 16n
287 If the \fbGLOB_KEEPSTAT\fr flag was set, \fbgl_statv\fr contains a
288 pointer to a null-terminated list of matched \fbstat\fr(2)
289 objects corresponding to the paths in \fbgl_pathc\fr.
290 .RE
291 .sp
292 .LP
293 If \fbglob()\fr terminates due to an error, it sets \fberrno\fr and
294 returns one of the following non-zero constants. defined in <\fbglob.h\fr>:
295 .sp
296 The following values are returned by \fbglob()\fr:
297 .sp
298 .ne 2
299 .na
300 \fb\fbGLOB_ABORTED\fr\fr
301 \fb\fbO\fr\fr
302 .ad
303 .RS 16n
304 The scan was stopped because \fbGLOB_ERR\fr was set or
305 \fb(\fr\fi*errfunc\fr\fb)\fr returned non-zero.
306 .RS 12n
307 Successful completion. The argument \figglob\{mi>\fr\fbgl_pathc\fr returns the
308 number of matched path names and the argument \figglob\{mi>\fr\fbgl_pathv\fr
309 contains a pointer to a null-terminated list of matched and sorted path names.
310 However, if \figglob\{mi>\fr\fbgl_pathc\fr is 0, the content of
311 \figglob\{mi>\fr\fbgl_pathv\fr is undefined.
312 .RE
313 .sp
314 .ne 2
315 .na
316 \fb\fbGLOB_NOMATCH\fr\fr
317 \fb\fbnon-zero\fr\fr
318 .ad
319 .RS 16n
320 The pattern does not match any existing path name, and \fbGLOB_NOCHECK\fr was
321 not set in flags.
322 .RS 12n
323 An error has occurred. Non-zero constants are defined in <\fbglob.h\fr>. The
324 arguments \figglob\{mi>\fr\fbgl_pathc\fr and \figglob\{mi>\fr\fbgl_pathv\fr are
325 still set as defined above.
326 .RE
327 .sp
328 .ne 2
329 .na
330 \fb\fbGLOB_NOSPACE\fr\fr
331 .ad
332 .RS 16n
333 An attempt to allocate memory failed.
334 .RE

```

```

487 .sp
488 .ne 2
489 .na
490 \fB\fBGLOB_NOSYS\fR\fR
491 .ad
492 .RS 16n
493 The requested function is not supported by this version of
494 \fBglob()\fR.
495 .RE

497 .LP
498 The arguments \fIpglob(mi>\fR\fBgl_pathc\fR and \fIpglob(mi>\fR\fBgl_pathv\fR
499 specified above.
500 .sp
501 .LP
502 The \fBglobfree()\fR function returns no value.
503 .SH USAGE
504 .sp
505 .LP
506 This function is not provided for the purpose of enabling utilities to perform
507 path name expansion on their arguments, as this operation is performed by the
508 shell, and utilities are explicitly not expected to redo this. Instead, it is
509 provided for applications that need to do path name expansion on strings
510 obtained from other sources, such as a pattern typed by a user or read from a
511 file.
512 .sp
513 .LP
514 If a utility needs to see if a path name matches a given pattern, it can use
515 \fBfnmatch\fR(3C).
516 .sp
517 .LP
518 Note that \fBgl_pathc\fR and \fBgl_pathv\fR have meaning even if \fBglob()\fR
519 fails. This allows \fBglob()\fR to report partial results in the event of an
520 error. However, if \fBgl_pathc\fR is 0, \fBgl_pathv\fR is unspecified even if
521 \fBglob()\fR did not return an error.
522 .sp
523 .LP
524 The \fBGLOB_NOCHECK\fR option could be used when an application wants to expand
525 a path name if wildcards are specified, but wants to treat the pattern as just
526 a string otherwise.
527 .sp
528 .LP
529 The new path names generated by a subsequent call with \fBGLOB_APPEND\fR are
530 not sorted together with the previous path names. This mirrors the way that the
531 shell handles path name expansion when multiple expansions are done on a
532 command line.
533 .sp
534 .LP
535 Applications that need tilde and parameter expansion should use the
536 \fBwordexp\fR(3C) function.
537 .SH EXAMPLES
538 .LP
539 \fBExample 1 \fRExample of \fBglob_doofs\fR function.
540 .sp
541 .LP
542 One use of the \fBGLOB_DOOFFS\fR flag is by applications that build an argument
543 list for use with the \fBexecv()\fR, \fBexecve()\fR, or \fBexecvp()\fR
544 functions (see \fBexec\fR(2)). Suppose, for example, that an application wants
545 to do the equivalent of:

547 .sp
548 .in +2
549 .nf
550 \fBls\fR \fB-l\fR *.c
551 .fi
552 .in -2

```

```

554 .sp
555 .LP
556 but for some reason:

558 .sp
559 .in +2
560 .nf
561 system("ls -l *.c")
562 .fi
563 .in -2

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

570 .sp
571 .in +2
572 .nf
573 globbuf.gl_offs = 2;
574 glob ("*.c", GLOB_DOOFFS, NULL, &globbuf);
575 globbuf.gl_pathv[0] = "ls";
576 globbuf.gl_pathv[1] = "-l";
577 execvp ("ls", &globbuf.gl_pathv[0]);
578 .fi
579 .in -2

581 .sp
582 .LP
583 Using the same example:

585 .sp
586 .in +2
587 .nf
588 \fBls\fR \fB-l\fR *.c *.h
589 .fi
590 .in -2

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

596 .sp
597 .in +2
598 .nf
599 \fBglobbuf.gl_offs = 2;
600 glob ("*.c", GLOB_DOOFFS, NULL, &globbuf);
601 glob ("*.h", GLOB_DOOFFS|GLOB_APPEND, NULL, &globbuf);
602 \&.\|.\.\fR
603 .fi
604 .in -2

606 .SH ATTRIBUTES
607 .sp
608 .LP
609 See \fBattributes\fR(5) for descriptions of the following attributes:
610 .sp

612 .sp
613 .TS
614 box;
615 c | c
616 l | l .
617 ATTRIBUTE TYPE ATTRIBUTE VALUE
618 _

```

```
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)
```