

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

1

```
*****
27029 Mon Jan 14 16:34:21 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-qnx systems */
400 /* #undef MISSING_NFDBITS */

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

```

457 #define HAVE_CRYPT_H 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

```

523 #define HAVE_GETRUSAGE 1

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

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

531 /* Define to 1 if you have the '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_systm.h> header file. */
661 #define HAVE_NETINET_IN_SYSTM_H 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

```

721 #define HAVE_SETENV 1

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

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

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

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

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

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

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

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

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

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

753 /* Define to 1 if you have the 'setutent' function. */
754 #define HAVE_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 Mon Jan 14 16:34:23 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 */
62 */
63 /*
64  * Copyright 1985, 1992 by Mortice Kern Systems Inc.  All rights reserved.
65  */

67 #ifndef _GLOB_H
68 #define _GLOB_H

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

70 #include <sys/feature_tests.h>
71 #include <sys/types.h>
72 #include <sys/stat.h>
73 #include <dirent.h>

75 #ifdef __cplusplus
76 extern "C" {
77 #endif

79 struct stat;

81 typedef struct glob_t {
82     /* Members required by POSIX */
83     size_t gl_pathc;      /* Total count of paths matched by pattern */
84     size_t gl_pathv;     /* Count of paths matched by pattern */
85     char **gl_pathv;     /* List of matched pathnames */
86     size_t gl_offs;      /* # of slots reserved in gl_pathv */
87     /* Non-POSIX extensions, from Openbsd */
88     int gl_matchc;       /* Count of paths matching pattern. */
89     int gl_flags;        /* Copy of flags parameter to glob. */
90     /* Members only accessed when Non-POSIX flags are specified. */
91     struct stat **gl_statv; /* Stat entries corresponding to gl_pathv */
92     /*
93     * Alternate filesystem access methods for glob; replacement
94     * versions of closedir(3), readdir(3), opendir(3), stat(2)
95     * and lstat(2).
96     */
97     void (*gl_closedir)(void *);
98     struct dirent *(*gl_readdir)(void *);
99     void *(*gl_opendir)(const char *);
100    int (*gl_lstat)(const char *, struct stat *);
101    int (*gl_stat)(const char *, struct stat *);
102    /* following are internal to the implementation */
103    char **gl_pathp;      /* gl_pathv + gl_offs */
104    int gl_pathn;        /* # of elements allocated */
105 } glob_t;

103 /*
104  * POSIX "flags" argument to glob function.
105  * "flags" argument to glob function.
106  */
107 #define GLOB_ERR      0x0001      /* Don't continue on directory error */
108 #define GLOB_MARK     0x0002      /* Mark directories with trailing / */
109 #define GLOB_NOSORT   0x0004      /* Don't sort pathnames */
110 #define GLOB_NOCHECK  0x0008      /* Return unquoted arg if no match */
111 #define GLOB_DOOFFS  0x0010      /* Ignore gl_offs unless set */
112 #define GLOB_APPEND   0x0020      /* Append to previous glob_t */
113 #define GLOB_NOESCAPE 0x0040      /* Backslashes do not quote M-chars */

114 /*
115  * Non-POSIX "flags" argument to glob function, from Openbsd.
116  */
117 #define GLOB_BRACE    0x0080      /* Expand braces ala csh. */
118 #define GLOB_MAGCHAR  0x0100      /* Pattern had globbing characters. */
119 #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 #ifndef __cplusplus
145 }
_____unchanged_portion_omitted_
```

```

*****
30190 Mon Jan 14 16:34:24 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  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright (c) 2012 Gary Mills
24  * Copyright 2008 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */
27 /*
28  * Copyright (c) 1989, 1993
29  * The Regents of the University of California. All rights reserved.
30  * This code is MKS code ported to Solaris originally with minimum
31  * modifications so that upgrades from MKS would readily integrate.
32  * The MKS basis for this modification was:
33  *
34  * This code is derived from software contributed to Berkeley by
35  * Guido van Rossum.
36  * $Id: glob.c 1.31 1994/04/07 22:50:43 mark
37  *
38  * Redistribution and use in source and binary forms, with or without
39  * modification, are permitted provided that the following conditions
40  * are met:
41  * 1. Redistributions of source code must retain the above copyright
42  * notice, this list of conditions and the following disclaimer.
43  * 2. Redistributions in binary form must reproduce the above copyright
44  * notice, this list of conditions and the following disclaimer in the
45  * documentation and/or other materials provided with the distribution.
46  * 3. Neither the name of the University nor the names of its contributors
47  * may be used to endorse or promote products derived from this software
48  * without specific prior written permission.
49  *
50  * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
51  * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
52  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
53  * ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
54  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
55  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
56  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
57  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
58  * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY

```

```

55 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
56 * SUCH DAMAGE.
57 * Additional modifications have been made to this code to make it
58 * 64-bit clean.
59 */

60 /*
61  * glob(3) -- a superset of the one defined in POSIX 1003.2.
62  * glob, globfree -- POSIX.2 compatible file name expansion routines.
63  *
64  * The [!...] convention to negate a range is supported (SysV, Posix, ksh).
65  * Copyright 1985, 1991 by Mortice Kern Systems Inc. All rights reserved.
66  *
67  * Optional extra services, controlled by flags not defined by POSIX:
68  *
69  * GLOB_QUOTE:
70  *   Escaping convention: \ inhibits any special meaning the following
71  *   character might have (except \ at end of string is retained).
72  * GLOB_MAGCHAR:
73  *   Set in gl_flags if pattern contained a globbing character.
74  * GLOB_NOMAGIC:
75  *   Same as GLOB_NOCHECK, but it will only append pattern if it did
76  *   not contain any magic characters. [Used in csh style globbing]
77  * GLOB_ALTDIRFUNC:
78  *   Use alternately specified directory access functions.
79  * GLOB_TILDE:
80  *   expand ~user/foo to the /home/dir/of/user/foo
81  * GLOB_BRACE:
82  *   expand {1,2}{a,b} to 1a 1b 2a 2b
83  * gl_matchc:
84  *   Number of matches in the current invocation of glob.
85  * Written by Eric Gisin.
86 */

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

90 #include <ctype.h>
91 #include <dirent.h>
92 #include <errno.h>
93 #include <glob.h>
94 #include <limits.h>
95 #include <pwd.h>
96 #pragma weak _glob = glob
97 #pragma weak _globfree = globfree

98 #include "lint.h"
99 #include <stdio.h>
100 #include <unistd.h>
101 #include <limits.h>
102 #include <stdlib.h>
103 #include <string.h>
104 #include <unistd.h>
105 #include <wchar.h>
106 #include <wctype.h>
107 #include <dirent.h>
108 #include <sys/stat.h>
109 #include <glob.h>
110 #include <errno.h>
111 #include <fnmatch.h>

112 #define DOLLAR '$'
113 #define DOT '.'
114 #define EOS '\0'
115 #define LBRACKET '['

```

```

104 #define NOT      '!'
105 #define QUESTION '?'
106 #define QUOTE    '\\\
107 #define RANGE   '-\
108 #define RBRACKET ']'
109 #define SEP      '/\
110 #define STAR     '**\
111 #define TILDE    '~\
112 #define UNDERSCORE '_\
113 #define LBRACE   '{\
114 #define RBRACE   '}'\
115 #define SLASH    '/\
116 #define COMMA    ',\
117 #define COLON    ':'\
63 #define GLOB__CHECK 0x80 /* stat generated paths */

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

122 typedef struct Char_s {
123     wchar_t wc;
124     uint_t at;
125 } Char;
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 *);

127 #define M_ALL      '**' /* Plus M_QUOTE */
128 #define M_END      ']' /* Plus M_QUOTE */
129 #define M_NOT      '!' /* Plus M_QUOTE */
130 #define M_ONE      '?' /* Plus M_QUOTE */
131 #define M_RNG     '-\
132 #define M_SET     '[' /* Plus M_QUOTE */
133 #define M_CLASS   ':' /* Plus M_QUOTE */
134 #define ismeta(c) (((c).at&M_QUOTE) != 0)

136 #define GLOB_LIMIT_MALLOC 65536
137 #define GLOB_LIMIT_STAT 2048
138 #define GLOB_LIMIT_READDIR 16384

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

143 struct glob_lim {
144     size_t glim_malloc;
145     size_t glim_stat;
146     size_t glim_readdir;
147 };

149 struct glob_path_stat {
150     char *gps_path;
151     struct stat *gps_stat;
152 };

154 static int compare(const void *, const void *);
155 static int compare_gps(const void *, const void *);
156 static int g_Ctoc(const Char *, char *, uint_t);
157 static int g_lstat(Char *, struct stat *, glob_t *);
158 static int *g_opendir(Char *, glob_t *);
159 static Char *g_strchr(const Char *, wchar_t);
160 static int g_stat(Char *, struct stat *, glob_t *);
161 static int glob0(const Char *, glob_t *, struct glob_lim *,

```

```

162     int (*)(const char *, int));
163 static int glob1(Char *, Char *, glob_t *, struct glob_lim *,
164 int (*)(const char *, int));
165 static int glob2(Char *, Char *, Char *, Char *, Char *,
166 glob_t *, struct glob_lim *,
167 int (*)(const char *, int));
168 static int glob3(Char *, Char *, Char *, Char *, Char *,
169 Char *, Char *, glob_t *, struct glob_lim *,
170 int (*)(const char *, int));
171 static int globextend(const Char *, glob_t *, struct glob_lim *,
172 struct stat *);
173 static
174 const Char *globtilde(const Char *, Char *, size_t, glob_t *);
175 static int globexpl(const Char *, glob_t *, struct glob_lim *,
176 int (*)(const char *, int));
177 static int globexp2(const Char *, const Char *, glob_t *,
178 struct glob_lim *, int (*)(const char *, int));
179 static int match(Char *, Char *, Char *, int);
180 #ifdef DEBUG
181 static void qprintf(const char *, Char *);
182 #endif

184 int
185 glob(const char *pattern, int flags, int (*errfunc)(const char *, int),
186 glob_t *pglob)
187 {
188     const char *patnext;
189     size_t n;
190     wchar_t c;
191     Char *bufnext, *bufend, patbuf[MAXPATHLEN];
192     struct glob_lim limit = { 0, 0, 0 };

194     if (strlen(pattern, PATH_MAX) == PATH_MAX)
195         return (GLOB_NOMATCH);

197     patnext = pattern;
198     if (!(flags & GLOB_APPEND)) {
199         pglob->gl_pathc = 0;
200         pglob->gl_pathv = NULL;
201         if ((flags & GLOB_KEEPCONST) != 0)
202             pglob->gl_statv = NULL;
203         if (!(flags & GLOB_DOOFFS))
204             pglob->gl_offs = 0;
205     }
206     pglob->gl_flags = flags & ~GLOB_MAGCHAR;
207     pglob->gl_matchc = 0;

209     if (pglob->gl_offs < 0 || pglob->gl_pathc < 0 ||
210         pglob->gl_offs >= INT_MAX || pglob->gl_pathc >= INT_MAX ||
211         pglob->gl_pathc >= INT_MAX - pglob->gl_offs - 1)
212         return (GLOB_NOSPACE);

214     bufnext = patbuf;
215     bufend = bufnext + MAXPATHLEN - 1;
216     if (flags & GLOB_NOESCAPE) {
217         while (bufnext < bufend) {
218             if ((n = mbtowlc(&c, patnext, PATH_MAX)) > 0) {
219                 patnext += n;
220                 bufnext->at = 0;
221                 (bufnext++)->wc = c;
222             } else if (n == 0) {
223                 break;
224             } else {
225                 return (GLOB_NOMATCH);
226             }
227         }

```

```

228     } else {
229         /* Protect the quoted characters. */
230         while (bufnext < bufend) {
231             if ((n = mbtowl(&c, patnext, PATH_MAX)) > 0) {
232                 patnext += n;
233                 if (c == QUOTE) {
234                     n = mbtowl(&c, patnext, PATH_MAX);
235                     if (n < 0)
236                         return (GLOB_NOMATCH);
237                     if (n > 0)
238                         patnext += n;
239                     if (n == 0)
240                         c = QUOTE;
241                     bufnext->at = M_PROTECT;
242                     (bufnext++)->wc = c;
243                 } else {
244                     bufnext->at = 0;
245                     (bufnext++)->wc = c;
246                 }
247             } else if (n == 0) {
248                 break;
249             } else {
250                 return (GLOB_NOMATCH);
251             }
252         }
253     }
254     bufnext->at = 0;
255     bufnext->wc = EOS;
256
257     if (flags & GLOB_BRACE)
258         return (globexpl(patbuf, pglob, &limit, errfunc));
259     else
260         return (glob0(patbuf, pglob, &limit, errfunc));
261 }
262
263 /*
264  * Expand recursively a glob {} pattern. When there is no more expansion
265  * invoke the standard globbing routine to glob the rest of the magic
266  * characters
267  * Free all space consumed by glob.
268  */
269 static int
270 globexpl(const Char *pattern, glob_t *pglob, struct glob_lim *limitp,
271          int (*errfunc)(const char *, int))
272 {
273     void
274     globfree(glob_t *gp)
275     {
276         const Char* ptr = pattern;
277         size_t i;
278
279         /* Protect a single {}, for find(1), like csh */
280         if (pattern[0].wc == LBRACE && pattern[1].wc == RBRACE &&
281             pattern[2].wc == EOS)
282             return (glob0(pattern, pglob, limitp, errfunc));
283         if (gp->gl_pathv == 0)
284             return;
285
286         if ((ptr = (const Char *) g_strchr(ptr, LBRACE)) != NULL)
287             return (globexp2(ptr, pattern, pglob, limitp, errfunc));
288         for (i = gp->gl_offs; i < gp->gl_offs + gp->gl_pathc; ++i)
289             free(gp->gl_pathv[i]);
290         free((void *)gp->gl_pathv);
291
292         return (glob0(pattern, pglob, limitp, errfunc));
293     }
294     gp->gl_pathc = 0;
295     gp->gl_pathv = NULLCPP;

```

```

283 }
284
285 /*
286  * Recursive brace globbing helper. Tries to expand a single brace.
287  * If it succeeds then it invokes globexpl with the new pattern.
288  * If it fails then it tries to glob the rest of the pattern and returns.
289  * Do filename expansion.
290  */
291 static int
292 globexp2(const Char *ptr, const Char *pattern, glob_t *pglob,
293          struct glob_lim *limitp, int (*errfunc)(const char *, int))
294 {
295     int
296     glob(const char *pattern, int flags,
297          int (*errfn)(const char *, int), glob_t *gp)
298     {
299         int i, rv;
300         Char *lm, *ls;
301         const Char *pe, *pm, *pl;
302         Char patbuf[MAXPATHLEN];
303         int rv;
304         size_t i;
305         size_t ipathc;
306         char *path;
307
308         /* copy part up to the brace */
309         for (lm = patbuf, pm = pattern; pm != ptr; *lm++ = *pm++)
310             ;
311         lm->at = 0;
312         lm->wc = EOS;
313         ls = lm;
314         if ((flags & GLOB_DOOFFS) == 0)
315             gp->gl_offs = 0;
316
317         /* Find the balanced brace */
318         for (i = 0, pe = ++ptr; pe->wc != EOS; pe++)
319             if (pe->wc == LBRACKET) {
320                 /* Ignore everything between [] */
321                 for (pm = pe++; pe->wc != RBRACKET &&
322                     pe->wc != EOS; pe++)
323                     ;
324                 if (pe->wc == EOS) {
325                     /*
326                      * We could not find a matching RBRACKET.
327                      * Ignore and just look for RBRACE
328                      */
329                     pe = pm;
330                 }
331             } else if (pe->wc == LBRACE) {
332                 i++;
333             } else if (pe->wc == RBRACE) {
334                 if (i == 0)
335                     break;
336                 i--;
337             }
338         if (!(flags & GLOB_APPEND)) {
339             gp->gl_pathc = 0;
340             gp->gl_pathn = gp->gl_offs + INITIAL;
341             gp->gl_pathv = (char **)malloc(sizeof(char *) * gp->gl_pathn);
342
343             /* Non matching braces; just glob the pattern */
344             if (i != 0 || pe->wc == EOS)
345                 return (glob0(patbuf, pglob, limitp, errfunc));
346             if (gp->gl_pathv == NULLCPP)
347                 return (GLOB_NOSPACE);
348             gp->gl_pathp = gp->gl_pathv + gp->gl_offs;

```

```

333     for (i = 0, pl = pm = ptr; pm <= pe; pm++) {
334         switch (pm->wc) {
335             case LBRACKET:
336                 /* Ignore everything between [] */
337                 for (pl = pm++; pm->wc != RBRACKET && pm->wc != EOS;
338                     pm++)
339                     ;
340                 if (pm->wc == EOS) {
341                     /*
342                      * We could not find a matching RBRACKET.
343                      * Ignore and just look for RBRACE
344                      */
345                     pm = pl;
346                 }
347                 for (i = 0; i < gp->gl_offs; ++i)
348                     gp->gl_pathv[i] = NULL;
349             case LBRACE:
350                 i++;
351                 break;
352             case RBRACE:
353                 if (i) {
354                     i--;
355                     break;
356                 }
357                 /* FALLTHROUGH */
358             case COMMA:
359                 if (i && pm->wc == COMMA)
360                     break;
361                 else {
362                     /* Append the current string */
363                     for (lm = ls; (pl < pm); *lm++ = *pl++)
364                         ;
365                     ipathc = gp->gl_pathc;
366                     rv = globit(0, pattern, gp, flags, errfn, &path);
367                 }
368                 if (rv == GLOB_ABORTED) {
369                     /*
370                      * Append the rest of the pattern after the
371                      * closing brace
372                      * User's error function returned non-zero, or GLOB_ERR was
373                      * set, and we encountered a directory we couldn't search.
374                      */
375                     for (pl = pe + 1;
376                         (*lm++ = *pl++).wc != EOS; /* */)
377                         ;
378                     /* Expand the current pattern */
379                     rv = globexpl(patbuf, pglob, limitp, errfunc);
380                     if (rv && rv != GLOB_NOMATCH)
381                         return (rv);
382                     /* move after the comma, to the next string */
383                     pl = pm + 1;
384                 }
385                 free(path);
386                 return (GLOB_ABORTED);
387             }
388         }
389     }
390     break;
391 }
392
393 default:
394     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);
140     }
141     return (0);
142 }
143
144 /*
145  * expand tilde from the passwd file.
146  */
147 static const Char *
148 globtilde(const Char *pattern, Char *patbuf, size_t patbuf_len, glob_t *pglob)
149 {
150     struct passwd *pwd;
151     char *h;
152     const Char *p;
153     Char *b, *eb, *q;
154     size_t n;
155     wchar_t c;
156
157     if (pattern->wc != TILDE || !(pglob->gl_flags & GLOB_TILDE))
158         return (pattern);
159
160     /* Copy up to the end of the string or / */
161     eb = &patbuf[patbuf_len - 1];
162     for (p = pattern + 1, q = patbuf;
163         q < eb && p->wc != EOS && p->wc != SLASH; *q++ = *p++)
164         ;
165
166     q->at = 0;
167     q->wc = EOS;
168
169     /* What to do if patbuf is full? */
170
171     if (patbuf[0].wc == EOS) {
172         /*
173          * handle a plain ~ or ~/ by expanding $HOME
174          * first and then trying the password file
175          */
176         if (issetugid() != 0 || (h = getenv("HOME")) == NULL) {
177             if ((pwd = getpwuid(getuid())) == NULL)
178                 return (pattern);
179         }
180         if (i == 0) {
181             if (flags & GLOB_NOCHECK)
182                 (void) append(gp, pattern);
183             else
184                 h = pwd->pw_dir;
185             rv = GLOB_NOMATCH;
186         }
187     } else {
188         /*
189          * Expand a ~user
190          */
191         if ((pwd = getpwnam((char *) patbuf)) == NULL)
192             return (pattern);
193         else
194             h = pwd->pw_dir;
195     }
196     gp->gl_pathp[gp->gl_pathc] = NULL;
197     free(path);
198
199     /* Copy the home directory */

```

```

443     for (b = patbuf; b < eb && *h != EOS; b++) {
444         if ((n = mbtowc(&c, h, PATH_MAX)) > 0) {
445             h += n;
446             b->at = 0;
447             b->wc = c;
448         } else {
449             break;
450         }
451     }

453     /* Append the rest of the pattern */
454     while (b < eb && (*b++ = *p++) != EOS)
455         ;
456     b->at = 0;
457     b->wc = EOS;

459     return (patbuf);
150     return (rv);
460 }

462 static int
463 g_charclass(const Char **patternp, Char **bufnextp)
464 {
465     const Char *pattern = *patternp + 1;
466     Char *bufnext = *bufnextp;
467     const Char *colon;
468     char cbuf[MB_LEN_MAX + 32];
469     wctype_t cc;
470     size_t len;

472     if ((colon = g_strchr(pattern, COLON)) == NULL ||
473         colon[1].wc != RBRACKET)
474         return (1); /* not a character class */

476     len = (size_t)(colon - pattern);
477     if (len + MB_LEN_MAX + 1 > sizeof (cbuf))
478         return (-1); /* invalid character class */
479     {
480         wchar_t w;
481         const Char *s1 = pattern;
482         char *s2 = cbuf;
483         size_t n = len;

485         /* Copy the string. */
486         while (n > 0) {
487             w = (s1++)->wc;
488             /* Character class names must be ASCII. */
489             if (iswascii(w)) {
490                 n--;
491                 *s2++ = w;
492             } else {
493                 return (-1); /* invalid character class */
494             }
495         }
496         *s2 = EOS;
497     }
498     if ((cc = wctype(cbuf)) == 0)
499         return (-1); /* invalid character class */
500     bufnext->at = M_QUOTE;
501     (bufnext++)->wc = M_CLASS;
502     bufnext->at = 0;
503     (bufnext++)->wc = cc;
504     *bufnextp = bufnext;
505     *patternp += len + 3;

507     return (0);

```

```

508 }

510 /*
511  * The main glob() routine: compiles the pattern (optionally processing
512  * quotes), calls glob1() to do the real pattern matching, and finally
513  * sorts the list (unless unsorted operation is requested). Returns 0
514  * if things went well, nonzero if errors occurred. It is not an error
515  * to find no matches.
516  * Recursive routine to match glob pattern, and walk directories.
517 */
518 static int
519 glob0(const Char *pattern, glob_t *pglob, struct glob_lim *limitp,
520        int (*errfunc)(const char *, int))
521 {
522     const Char *qpatnext;
523     int err, oldpathc;
524     wchar_t c;
525     int a;
526     Char *bufnext, patbuf[MAXPATHLEN];
527     size_t n;
528     size_t m;
529     ssize_t end = 0; /* end of expanded directory */
530     char *pat = (char *)sp; /* pattern component */
531     char *dp = (*path) + dend;
532     int expand = 0; /* path has pattern */
533     char *cp;
534     struct stat64 sb;
535     DIR *dirp;
536     struct dirent64 *d;
537     int err;

539     qpatnext = globtilde(pattern, patbuf, MAXPATHLEN, pglob);
540     oldpathc = pglob->gl_pathc;
541     bufnext = patbuf;

543     /* We don't need to check for buffer overflow any more. */
544     while ((a = qpatnext->at), (c = (qpatnext++)->wc) != EOS) {
545         switch (c) {
546             case LBRACKET:
547                 if (a != 0) {
548                     bufnext->at = a;
549                     (bufnext++)->wc = c;
550                     break;
551                 }
552                 for (;;)
553                     switch (*dp++ = *(unsigned char *)sp++) {
554                         case '\0': /* end of source path */
555                             if (expand)
556                                 goto Expand;
557                             else {
558                                 if (!(flags & GLOB_NOCHECK) ||
559                                     flags & (GLOB_CHECK|GLOB_MARK))
560                                     if (stat64(*path, &sb) < 0) {
561                                         return (0);
562                                     }
563                             }
564                         }
565                 a = qpatnext->at;
566                 c = qpatnext->wc;
567                 if (a == 0 && c == NOT)
568                     ++qpatnext;
569                 if (qpatnext->wc == EOS ||
570                     g_strchr(qpatnext+1, RBRACKET) == NULL) {
571                     bufnext->at = 0;
572                     (bufnext++)->wc = LBRACKET;
573                     if (a == 0 && c == NOT)

```

```

549         --qpatnext;
550         break;
184         if (flags & GLOB_MARK && S_ISDIR(sb.st_mode)) {
185             *dp = '\0';
186             *--dp = '/';
551     }
552     bufnext->at = M_QUOTE;
553     (bufnext++)->wc = M_SET;
554     if (a == 0 && c == NOT) {
555         bufnext->at = M_QUOTE;
556         (bufnext++)->wc = M_NOT;
557     }
558     a = qpatnext->at;
559     c = (qpatnext++)->wc;
560     do {
561         if (a == 0 && c == LBRACKET &&
562             qpatnext->wc == COLON) {
563             do {
564                 err = g_charclass(&qpatnext,
565                     &bufnext);
566                 if (err)
567                     break;
568                 a = qpatnext->at;
569                 c = (qpatnext++)->wc;
570             } while (a == 0 && c == LBRACKET &&
571                 qpatnext->wc == COLON);
572             if (err == -1 &&
573                 !(pglob->gl_flags & GLOB_NOCHECK))
574                 return (GLOB_NOMATCH);
575             if (a == 0 && c == RBRACKET)
576                 break;
577         }
578         bufnext->at = a;
579         (bufnext++)->wc = c;
580         if (qpatnext->at == 0 &&
581             qpatnext->wc == RANGE) {
582             a = qpatnext[1].at;
583             c = qpatnext[1].wc;
584             if (qpatnext[1].at != 0 ||
585                 qpatnext[1].wc != RBRACKET) {
586                 bufnext->at = M_QUOTE;
587                 (bufnext++)->wc = M_RNG;
588                 bufnext->at = a;
589                 (bufnext++)->wc = c;
590                 qpatnext += 2;
591             }
592         }
593         a = qpatnext->at;
594         c = (qpatnext++)->wc;
595     } while (a != 0 || c != RBRACKET);
596     pglob->gl_flags |= GLOB_MAGCHAR;
597     bufnext->at = M_QUOTE;
598     (bufnext++)->wc = M_END;
599     break;
600 case QUESTION:
601     if (a != 0) {
602         bufnext->at = a;
603         (bufnext++)->wc = c;
604         break;
605     }
606     pglob->gl_flags |= GLOB_MAGCHAR;
607     bufnext->at = M_QUOTE;
608     (bufnext++)->wc = M_ONE;
609     break;
610 case STAR:
611     if (a != 0) {

```

```

612         bufnext->at = a;
613         (bufnext++)->wc = c;
614         break;
615     }
616     pglob->gl_flags |= GLOB_MAGCHAR;
617     /*
618     * collapse adjacent stars to one,
619     * to avoid exponential behavior
620     */
621     if (bufnext == patbuf ||
622         bufnext[-1].at != M_QUOTE ||
623         bufnext[-1].wc != M_ALL) {
624         bufnext->at = M_QUOTE;
625         (bufnext++)->wc = M_ALL;
626     }
627     break;
628 default:
629     bufnext->at = a;
630     (bufnext++)->wc = c;
631     break;
632 }
633 }
634 bufnext->at = 0;
635 bufnext->wc = EOS;
636 #ifndef DEBUG
637     qprintf("glob0:glob1:patbuf", patbuf);
638 #endif
639
640 if ((err = glob1(patbuf, patbuf+MAXPATHLEN-1, pglob, limitp, errfunc)
641     != 0)
642     return (err);
643
644 /*
645  * If there was no match we are going to append the pattern
646  * if GLOB_NOCHECK was specified or if GLOB_NOMAGIC was specified
647  * and the pattern did not contain any magic characters
648  * GLOB_NOMAGIC is there just for compatibility with csh.
649  */
650 if (pglob->gl_pathc == oldpathc) {
651     if ((pglob->gl_flags & GLOB_NOCHECK) ||
652         ((pglob->gl_flags & GLOB_NOMAGIC) &&
653         !(pglob->gl_flags & GLOB_MAGCHAR)))
654         return (globextend(pattern, pglob, limitp, NULL));
655     else
656         return (GLOB_NOMATCH);
657 }
658 if (!(pglob->gl_flags & GLOB_NOSORT)) {
659     if ((pglob->gl_flags & GLOB_KEEPSTAT)) {
660         /* Keep the paths and stat info synced during sort */
661         struct glob_path_stat *path_stat;
662         int i;
663         int n = pglob->gl_pathc - oldpathc;
664         int o = pglob->gl_offs + oldpathc;
665
666         if ((path_stat = calloc(n, sizeof (*path_stat))) ==
667             NULL)
668             if (append(gp, *path) < 0) {
188                 return (GLOB_NOSPACE);
668             }
669         for (i = 0; i < n; i++) {
670             path_stat[i].gps_path = pglob->gl_pathv[o + i];
671             path_stat[i].gps_stat = pglob->gl_statv[o + i];
672         }
673         qsort(path_stat, n, sizeof (*path_stat), compare_gps);
674         for (i = 0; i < n; i++) {
675             pglob->gl_pathv[o + i] = path_stat[i].gps_path;
676             pglob->gl_statv[o + i] = path_stat[i].gps_stat;

```



```

677     }
678     } else {
679         free(path_stat);
680     }
681     qsort(pglob->gl_pathv + pglob->gl_offs + oldpathc,
682         pglob->gl_pathc - oldpathc, sizeof (char *),
683         compare);
684     }
685     return (0);
686 }
687
688 static int
689 compare(const void *p, const void *q)
690 {
691     return (strcmp(*(char **)p, *(char **)q));
692 }
693
694 static int
695 compare_gps(const void *_p, const void *_q)
696 {
697     const struct glob_path_stat *p = (const struct glob_path_stat *)_p;
698     const struct glob_path_stat *q = (const struct glob_path_stat *)_q;
699     case '/':
700         if (expand)
701             goto Expand;
702         end = dp - *path;
703         pat = (char *)sp;
704         break;
705
706     return (strcmp(p->gps_path, q->gps_path));
707 }
708
709 static int
710 glob1(Char *pattern, Char *pattern_last, glob_t *pglob,
711     struct glob_lim *limitp, int (*errfunc)(const char *, int))
712 {
713     Char pathbuf[MAXPATHLEN];
714
715     /* A null pathname is invalid -- POSIX 1003.1 sect. 2.4. */
716     if (pattern->wc == EOS)
717         return (0);
718     return (glob2(pathbuf, pathbuf+MAXPATHLEN-1,
719         pathbuf, pathbuf+MAXPATHLEN-1,
720         pattern, pattern_last, pglob, limitp, errfunc));
721 }
722
723 /*
724 * The functions glob2 and glob3 are mutually recursive; there is one level
725 * of recursion for each segment in the pattern that contains one or more
726 * meta characters.
727 */
728 static int
729 glob2(Char *pathbuf, Char *pathbuf_last, Char *pathend, Char *pathend_last,
730     Char *pattern, Char *pattern_last, glob_t *pglob,
731     struct glob_lim *limitp, int (*errfunc)(const char *, int))
732 {
733     struct stat sb;
734     Char *p, *q;

```

```

729     int anymeta;
730
731     /*
732     * Loop over pattern segments until end of pattern or until
733     * segment with meta character found.
734     */
735     for (anymeta = 0; ; ) {
736         if (pattern->wc == EOS) {
737             /* End of pattern? */
738             pathend->at = 0;
739             pathend->wc = EOS;
740
741             if ((pglob->gl_flags & GLOB_LIMIT) &&
742                 limitp->glim_stat++ >= GLOB_LIMIT_STAT) {
743                 errno = 0;
744                 pathend->at = 0;
745                 (pathend++)->wc = SEP;
746                 pathend->at = 0;
747                 pathend->wc = EOS;
748                 return (GLOB_NOSPACE);
749             }
750             Expand:
751             /* determine directory and open it */
752             (*path)[end] = '\0';
753             dirp = opendir(*path == '\0' ? "." : *path);
754             if (dirp == NULL) {
755                 if (errno != 0 && errno(*path, errno) != 0 ||
756                     flags & GLOB_ERR) {
757                     return (GLOB_ABORTED);
758                 }
759                 if (g_lstat(pathbuf, &sb, pglob))
760                     return (0);
761
762                 if (((pglob->gl_flags & GLOB_MARK) &&
763                     (pathend[-1].at != 0 ||
764                     pathend[-1].wc != SEP)) &&
765                     (S_ISDIR(sb.st_mode) ||
766                     (S_ISLNK(sb.st_mode) &&
767                     (g_stat(pathbuf, &sb, pglob) == 0) &&
768                     S_ISDIR(sb.st_mode)))) {
769                     if (pathend+1 > pathend_last)
770                         return (GLOB_NOSPACE);
771                     pathend->at = 0;
772                     (pathend++)->wc = SEP;
773                     pathend->at = 0;
774                     pathend->wc = EOS;
775                 }
776                 ++pglob->gl_matchc;
777                 return (globextend(pathbuf, pglob, limitp, &sb));
778             }
779
780             /* Find end of next segment, copy tentatively to pathend. */
781             q = pathend;
782             p = pattern;
783             while (p->wc != EOS && p->wc != SEP) {
784                 if (ismeta(*p))
785                     anymeta = 1;
786                 if (q+1 > pathend_last)
787                     /* extract pattern component */
788                     n = sp - pat;
789                 if ((cp = malloc(n)) == NULL) {
790                     (void) closedir(dirp);
791                     return (GLOB_NOSPACE);
792                 }
793                 *q++ = *p++;
794
795                 pat = memcpy(cp, pat, n);
796                 pat[n-1] = '\0';
797                 if (*--sp != '\0')

```

```

230         flags |= GLOB_CHECK;

781     if (!anymeta) {          /* No expansion, do next segment. */
782         pathend = q;
783         pattern = p;
784         while (pattern->wc == SEP) {
785             if (pathend+1 > pathend_last)
232             /* expand path to max. expansion */
233             n = dp - *path;
234             *path = realloc(*path,
235                 strlen(*path) + NAME_MAX + strlen(sp) + 1);
236             if (*path == NULL) {
237                 (void) closedir(dirp);
238                 free(pat);
786                 return (GLOB_NOSPACE);
787             *pathend++ = *pattern++;
788         }
789     } else {
790         /* Need expansion, recurse. */
791         return (glob3(pathbuf, pathbuf_last, pathend,
792             pathend_last, pattern, p, pattern_last,
793             pglob, limitp, errfunc));
794     }
795 }
796 /* NOTREACHED */
797 }
241     dp = (*path) + n;

799 static int
800 glob3(Char *pathbuf, Char *pathbuf_last, Char *pathend, Char *pathend_last,
801 Char *pattern, Char *restpattern, Char *restpattern_last, glob_t *pglob,
802 struct glob_lim *limitp, int (*errfunc)(const char *, int))
803 {
804     struct dirent *dp;
805     DIR *dirp;
806     int err;
807     char buf[MAXPATHLEN];

809     /*
810     * The readdirfunc declaration can't be prototyped, because it is
811     * assigned, below, to two functions which are prototyped in glob.h
812     * and dirent.h as taking pointers to differently typed opaque
813     * structures.
814     */
815     struct dirent *(*readdirfunc)(void *);

817     if (pathend > pathend_last)
818         return (GLOB_NOSPACE);
819     pathend->at = 0;
820     pathend->wc = EOS;
821     errno = 0;

823     if ((dirp = g_opendir(pathbuf, pglob)) == NULL) {
824         /* TODO: don't call for ENOENT or ENOTDIR? */
825         if (errfunc) {
826             if (g_Ctoc(pathbuf, buf, sizeof (buf)))
827                 return (GLOB_ABORTED);
828             if (errfunc(buf, errno) ||
829                 pglob->gl_flags & GLOB_ERR)
830                 return (GLOB_ABORTED);
831         }
832     }
833     return (0);

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

```

```

837     /* Search directory for matching names. */
838     if (pglob->gl_flags & GLOB_ALTDIRFUNC)
839         readdirfunc = pglob->gl_readdir;
840     else
841         readdirfunc = (struct dirent *(*)(void *))readdir;
842     while ((dp = (*readdirfunc)(dirp)) != NULL) {
843         char *sc;
844         Char *dc;
845         size_t n;
846         wchar_t w;

848         if ((pglob->gl_flags & GLOB_LIMIT) &&
849             limitp->glim_readdir++ >= GLOB_LIMIT_READDIR) {
850             errno = 0;
851             pathend->at = 0;
852             (pathend++)->wc = SEP;
853             pathend->at = 0;
854             pathend->wc = EOS;
855             err = GLOB_NOSPACE;
856             break;
857         }

859         /* Initial DOT must be matched literally. */
860         if (dp->d_name[0] == DOT && pattern->wc != DOT)
245             while ((d = readdir64(dirp)) != NULL) {
246                 cp = d->d_name;
247                 if ((flags & GLOB_NOESCAPE)
248                     ? fnmatch(pat, cp, FNM_PERIOD/FNM_NOESCAPE)
249                     : fnmatch(pat, cp, FNM_PERIOD))
861                     continue;
862                 dc = pathend;
863                 sc = dp->d_name;
864                 while (dc < pathend_last) {
865                     if ((n = mbtowc(&w, sc, MB_LEN_MAX)) <= 0) {
866                         sc += 1;
867                         dc->at = 0;
868                         dc->wc = EOS;
869                     } else {
870                         sc += n;
871                         dc->at = 0;
872                         dc->wc = w;
873                     }
874                     dc++;
875                     if (n <= 0)
876                         break;
877                 }
878                 if (dc >= pathend_last) {
879                     dc->at = 0;
880                     dc->wc = EOS;
881                     err = GLOB_NOSPACE;
882                     break;
883                 }
884                 if (n < 0) {
885                     dc->at = 0;
886                     dc->wc = EOS;
887                     continue;
888                 }

890                 if (!match(pathend, pattern, restpattern, GLOB_LIMIT_RECUR)) {
891                     pathend->at = 0;
892                     pathend->wc = EOS;
893                     continue;
894                 }
895                 err = glob2(pathbuf, pathbuf_last, --dc, pathend_last,
896                     restpattern, restpattern_last, pglob, limitp,

```

```

897         errfunc);
898         if (err)
252             n = strlen(cp);
253             (void) memcpy((*path) + end, cp, n);
254             m = dp - *path;
255             err = globit(end+n, sp, gp, flags, errfn, path);
256             dp = (*path) + m; /* globit can move path */
257             if (err != 0)
899                 break;
900     }

902     if (pglob->gl_flags & GLOB_ALTDIRFUNC)
903         (*pglob->gl_closedir)(dirp);
904     else
905         closedir(dirp);
261         (void) closedir(dirp);
262         free(pat);
906     return (err);
264 }
265 /* NOTREACHED */
907 }

```

```

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

937     newn = 2 + pglob->gl_pathc + pglob->gl_offs;
938     if (pglob->gl_offs >= INT_MAX ||
939         pglob->gl_pathc >= INT_MAX ||
940         newn >= INT_MAX ||
941         SIZE_MAX / sizeof (*pathv) <= newn ||
942         SIZE_MAX / sizeof (*statv) <= newn) {
943     nospace:
944         for (i = pglob->gl_offs; i < (ssize_t)(newn - 2); i++) {
945             if (pglob->gl_pathv && pglob->gl_pathv[i])
946                 free(pglob->gl_pathv[i]);
947             if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
948                 pglob->gl_pathv && pglob->gl_pathv[i])
949                 free(pglob->gl_statv[i]);

```

```

950     }
951     if (pglob->gl_pathv) {
952         free(pglob->gl_pathv);
953         pglob->gl_pathv = NULL;
954     }
955     if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
956         pglob->gl_statv) {
957         free(pglob->gl_statv);
958         pglob->gl_statv = NULL;
959     }
960     return (GLOB_NOSPACE);
961 }

963     pathv = realloc(pglob->gl_pathv, newn * sizeof (*pathv));
964     if (pathv == NULL)
965         goto nospace;
966     if (pglob->gl_pathv == NULL && pglob->gl_offs > 0) {
967         /* first time around -- clear initial gl_offs items */
968         pathv += pglob->gl_offs;
969         for (i = pglob->gl_offs; --i >= 0; )
970             *--pathv = NULL;
971     }
972     pglob->gl_pathv = pathv;

974     if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0) {
975         statv = realloc(pglob->gl_statv, newn * sizeof (*statv));
976         if (statv == NULL)
977             goto nospace;
978         if (pglob->gl_statv == NULL && pglob->gl_offs > 0) {
979             /* first time around -- clear initial gl_offs items */
980             statv += pglob->gl_offs;
981             for (i = pglob->gl_offs; --i >= 0; )
982                 *--statv = NULL;
983         }
984         pglob->gl_statv = statv;
985         if (sb == NULL)
986             statv[pglob->gl_offs + pglob->gl_pathc] = NULL;
987     }
988     else {
989         limitp->glim_malloc += sizeof (**statv);
990         if ((pglob->gl_flags & GLOB_LIMIT) &&
991             limitp->glim_malloc >= GLOB_LIMIT_MALLOC) {
992             errno = 0;
993             return (GLOB_NOSPACE);
994         }
995         if ((statv[pglob->gl_offs + pglob->gl_pathc] =
996             malloc(sizeof (**statv))) == NULL)
997             goto copy_error;
998         memcpy(statv[pglob->gl_offs + pglob->gl_pathc], sb,
999             sizeof (*sb));
1000     }
1001     statv[pglob->gl_offs + pglob->gl_pathc + 1] = NULL;

1003     len = MB_LEN_MAX;
1004     p = path;
1005     while ((n = wctomb(junk, p->wc)) > 0) {
1006         len += n;
1007         if ((p++)->wc == EOS)
1008             break;
1009     }

1011     limitp->glim_malloc += len;
1012     if ((copy = malloc(len)) != NULL) {
1013         if (g_Ctoc(path, copy, len)) {
1014             free(copy);
1015             return (GLOB_NOSPACE);

```

```

1016     }
1017     pathv[pglob->gl_offs + pglob->gl_pathc++] = copy;
1018 }
1019 pathv[pglob->gl_offs + pglob->gl_pathc] = NULL;

1021 if ((pglob->gl_flags & GLOB_LIMIT) &&
1022     (newn * sizeof (*pathv)) + limitp->glim_malloc >
1023     GLOB_LIMIT_MALLOC) {
1024     errno = 0;
1025     return (GLOB_NOSPACE);
1026 }
1027 copy_error:
1028 return (copy == NULL ? GLOB_NOSPACE : 0);
1029 *
1030 return (strcoll(*(char **)npp1, *(char **)npp2));
1031 }

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

1044     if (recur-- == 0)
1045         return (1);
1046     if ((cp = malloc(strlen(str)+1)) == NULL)
1047         return (GLOB_NOSPACE);
1048     gp->gl_pathp[gp->gl_pathc++] = strcpy(cp, str);

1049     while (pat < patend) {
1050         c = *pat++;
1051         switch (c.wc) {
1052             case M_ALL:
1053                 if (c.at != M_QUOTE) {
1054                     k = *name++;
1055                     if (k.at != c.at || k.wc != c.wc)
1056                         return (0);
1057                     break;
1058                 }
1059                 if ((gp->gl_pathc + gp->gl_offs) >= gp->gl_pathn) {
1060                     gp->gl_pathn *= 2;
1061                     gp->gl_pathv = (char **)realloc((void *)gp->gl_pathv,
1062                                                     gp->gl_pathn * sizeof (char *));
1063                     if (gp->gl_pathv == NULLCPP)
1064                         return (GLOB_NOSPACE);
1065                     gp->gl_pathp = gp->gl_pathv + gp->gl_offs;
1066                 }
1067                 while (pat < patend && pat->at == M_QUOTE &&
1068                     pat->wc == M_ALL)
1069                     pat++; /* eat consecutive '*' */
1070                 if (pat == patend)
1071                     return (1);
1072                 do {
1073                     if (match(name, pat, patend, recur))
1074                         return (1);
1075                 } while ((name++)->wc != EOS);
1076                 return (0);
1077             case M_ONE:

```

```

1078         if (c.at != M_QUOTE) {
1079             k = *name++;
1080             if (k.at != c.at || k.wc != c.wc)
1081                 return (0);
1082             break;
1083         }
1084         if ((name++)->wc == EOS)
1085             return (0);
1086         break;
1087     case M_SET:
1088         if (c.at != M_QUOTE) {
1089             k = *name++;
1090             if (k.at != c.at || k.wc != c.wc)
1091                 return (0);
1092             break;
1093         }
1094         ok = 0;
1095         if ((k = *name++).wc == EOS)
1096             return (0);
1097         if ((negate_range = (pat->at == M_QUOTE &&
1098             pat->wc == M_NOT)) != 0)
1099             ++pat;
1100         while ((c = *pat++).at != M_QUOTE || c.wc != M_END) {
1101             if (c.at == M_QUOTE && c.wc == M_CLASS) {
1102                 Char cc;
1103
1104                 cc.at = pat->at;
1105                 cc.wc = pat->wc;
1106                 if (iswctype(k.wc, cc.wc))
1107                     ok = 1;
1108                 ++pat;
1109             }
1110             if (pat->at == M_QUOTE && pat->wc == M_RNG) {
1111                 if (c.wc <= k.wc && k.wc <= pat[1].wc)
1112                     ok = 1;
1113                 pat += 2;
1114             } else if (c.wc == k.wc)
1115                 ok = 1;
1116         }
1117         if (ok == negate_range)
1118             return (0);
1119         break;
1120     default:
1121         k = *name++;
1122         if (k.at != c.at || k.wc != c.wc)
1123             return (0);
1124         break;
1125     }
1126     return (name->wc == EOS);
1127 }

1128 /* Free allocated data belonging to a glob_t structure. */
1129 void
1130 globfree(glob_t *pglob)
1131 {
1132     int i;
1133     char **pp;

1134     if (pglob->gl_pathv != NULL) {
1135         pp = pglob->gl_pathv + pglob->gl_offs;
1136         for (i = pglob->gl_pathc; i--; ++pp)
1137             if (*pp)
1138                 free(*pp);
1139         free(pglob->gl_pathv);
1140         pglob->gl_pathv = NULL;

```

```

1132     }
1133     if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
1134         pglob->gl_statv != NULL) {
1135         for (i = 0; i < pglob->gl_pathc; i++) {
1136             if (pglob->gl_statv[i] != NULL)
1137                 free(pglob->gl_statv[i]);
1138         }
1139         free(pglob->gl_statv);
1140         pglob->gl_statv = NULL;
1141     }
1142 }

1144 static DIR *
1145 g_opendir(Char *str, glob_t *pglob)
1146 {
1147     char buf[MAXPATHLEN];

1149     if (str->wc == EOS)
1150         strcpy(buf, ".", sizeof (buf));
1151     else {
1152         if (g_Ctoc(str, buf, sizeof (buf)))
1153             return (NULL);
1154     }

1156     if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1157         return ((*pglob->gl_opendir)(buf));

1159     return (opendir(buf));
1160 }

1162 static int
1163 g_lstat(Char *fn, struct stat *sb, glob_t *pglob)
1164 {
1165     char buf[MAXPATHLEN];

1167     if (g_Ctoc(fn, buf, sizeof (buf)))
1168         return (-1);
1169     if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1170         return ((*pglob->gl_lstat)(buf, sb));
1171     return (lstat(buf, sb));
1172 }

1174 static int
1175 g_stat(Char *fn, struct stat *sb, glob_t *pglob)
1176 {
1177     char buf[MAXPATHLEN];

1179     if (g_Ctoc(fn, buf, sizeof (buf)))
1180         return (-1);
1181     if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1182         return ((*pglob->gl_stat)(buf, sb));
1183     return (stat(buf, sb));
1184 }

1186 static Char *
1187 g_strchr(const Char *str, wchar_t ch)
1188 {
1189     do {
1190         if (str->at == 0 && str->wc == ch)
1191             return ((Char *)str);
1192     } while ((str++)->wc != EOS);
1193     return (NULL);
1194 }

1196 static int
1197 g_Ctoc(const Char *str, char *buf, uint_t len)

```

```

1198 {
1199     int n;
1200     wchar_t w;

1202     while (len >= MB_LEN_MAX) {
1203         w = (str++)->wc;
1204         if ((n = wctomb(buf, w)) > 0) {
1205             len -= n;
1206             buf += n;
1207         }
1208         if (n < 0)
1209             break;
1210         if (w == EOS)
1211             return (0);
1212     }
1213     return (1);
1214 }

1216 #ifdef DEBUG
1217 static void
1218 qprintf(const char *str, Char *s)
1219 {
1220     Char *p;

1222     (void) printf("%s:\n", str);
1223     for (p = s; p->wc != EOS; p++)
1224         (void) printf("%wc", p->wc);
1225     (void) printf("\n");
1226     for (p = s; p->wc != EOS; p++)
1227         (void) printf("%c", p->at & M_PROTECT ? '\'' : ' ');
1228     (void) printf("\n");
1229     for (p = s; p->wc != EOS; p++)
1230         (void) printf("%c", ismeta(*p) ? '_' : ' ');
1231     (void) printf("\n");
1232 }
1233 #endif

```

```

*****
17719 Mon Jan 14 16:34:24 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>

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

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

```

```

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

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

101 .fi
102 .in -2

104 .sp
105 .LP
106 The \fBglob()\fR function stores the number of matched path names into
107 \fIpglob\<mi>\fR\fBglob_pathc\fR and a pointer to a list of pointers to path
108 names into \fIpglob\<mi>\fR\fBglob_pathv\fR. The path names are in sort order as
109 defined by the current setting of the \fBLC_COLLATE\fR category. The first
110 pointer after the last path name is a \fBNULL\fR pointer. If the pattern does
111 not match any path names, the returned number of matched paths is set to 0, and
112 the contents of \fIpglob\<mi>\fR\fBglob_pathv\fR are implementation-dependent.
113 .sp
114 .LP
115 It is the caller's responsibility to create the structure pointed to by
116 \fIpglob\fR. The \fBglob()\fR function allocates other space as needed,
117 including the memory pointed to by \fBglob_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 The following values are returned by \fbglob()\fr:
296 .sp
297 .ne 2
298 .na
299 \fb\fbGLOB_ABORTED\fr\fr
300 \fb\fbO\fr\fr
301 .ad
302 .RS 16n
303 The scan was stopped because \fbGLOB_ERR\fr was set or
304 \fb(\fr\fi*errfunc\fr\fb)\fr returned non-zero.
305 .RS 12n
306 Successful completion. The argument \figglob\{mi>\fr\fbgl_pathc\fr returns the
307 number of matched path names and the argument \figglob\{mi>\fr\fbgl_pathv\fr
308 contains a pointer to a null-terminated list of matched and sorted path names.
309 However, if \figglob\{mi>\fr\fbgl_pathc\fr is 0, the content of
310 \figglob\{mi>\fr\fbgl_pathv\fr is undefined.
311 .RE
312 .sp
313 .ne 2
314 .na
315 \fb\fbGLOB_NOMATCH\fr\fr
316 \fb\fbnon-zero\fr\fr
317 .ad
318 .RS 16n
319 The pattern does not match any existing path name, and \fbGLOB_NOCHECK\fr was
320 not set in flags.
321 .RS 12n
322 An error has occurred. Non-zero constants are defined in <\fbglob.h\fr>. The
323 arguments \figglob\{mi>\fr\fbgl_pathc\fr and \figglob\{mi>\fr\fbgl_pathv\fr are
324 still set as defined above.
325 .RE
326 .sp
327 .ne 2
328 .na
329 \fb\fbGLOB_NOSPACE\fr\fr
330 .ad
331 .RS 16n
332 An attempt to allocate memory failed.
333 .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)
```