```
**********************************************************
   27029 Mon Dec 17 10:05:15 2012
new/usr/src/cmd/ssh/include/config.h
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
/* config.h.  Generated by configure.  */
/* config.h.in.  Generated from configure.ac by autoheader.  */
/* $Id: acconfig.h,v 1.145 2002/09/26 00:38:48 tim Exp $ */

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

#ifndef _CONFIG_H
#define _CONFIG_H

#ifdef __cplusplus
extern "C" {
#endif


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


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

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

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

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

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

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

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

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

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

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

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

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

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



```
#define IP_TOS_IS_BROKEN 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
127 #define ALTPRIVSEP

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

390 /*
```

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

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

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

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

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

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



```
457 #define HAVE_CRYPT_H 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
523 #define HAVE_GETRUSAGE 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



```
589 #define HAVE_LIBDL 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
655 #define HAVE_NETDB_H 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



```
721 #define HAVE_SETENV 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
787 #define HAVE_STDLIB_H 1

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

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

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

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

 981 #ifdef __cplusplus
 982 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    5560 Mon Dec 17 10:05:16 2012
new/usr/src/head/glob.h
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

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

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



```
 */

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

#ifndef _GLOB_H
#define _GLOB_H

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

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

#ifdef  __cplusplus
extern "C" {
#endif

struct stat;

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

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

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

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

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

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

144 #ifdef  __cplusplus
145 }
```

*unchanged_portion_omitted_*

```
**********************************************************
     654 Mon Dec 17 10:05:17 2012
new/usr/src/lib/libc/port/regex/charclass.h
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
```

```
/*
 * Public domain, 2008, Todd C. Miller <Todd.Miller@courtesan.com>
 *
 * $OpenBSD: charclass.h,v 1.1 2008/10/01 23:04:13 millert Exp $
 */

/*
 * POSIX character class support for fnmatch() and glob().
 */
static struct cclass {
        const char *name;
        int (*isctype)(int);
} cclasses[] = {
        { "alnum",      isalnum },
        { "alpha",      isalpha },
        { "blank",      isblank },
        { "cntrl",      iscntrl },
        { "digit",      isdigit },
        { "graph",      isgraph },
        { "lower",      islower },
        { "print",      isprint },
        { "punct",      ispunct },
        { "space",      isspace },
        { "upper",      isupper },
        { "xdigit",     isxdigit },
        { NULL,         NULL }
};

#define NCCLASSES       (sizeof (cclasses) / sizeof (cclasses[0]) - 1)
```

```
**********************************************************
   30108 Mon Dec 17 10:05:18 2012
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()
**********************************************************
```

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

/*
 * Copyright (c) 2012 Gary Mills
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*      $OpenBSD: glob.c,v 1.39 2012/01/20 07:09:42 tedu Exp $ */
/*
 * Copyright (c) 1989, 1993
 *      The Regents of the University of California.  All rights reserved.
 * This code is MKS code ported to Solaris originally with minimum
 * modifications so that upgrades from MKS would readily integrate.
 * The MKS basis for this modification was:
 *
 * This code is derived from software contributed to Berkeley by
 * Guido van Rossum.
 *      $Id: glob.c 1.31 1994/04/07 22:50:43 mark
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
```



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

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

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

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

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

#include "charclass.h"
#define GLOB__CHECK     0x80    /* stat generated paths */

#define DOLLAR          '$'
#define DOT             '.'
```

```
#define EOS             '\0'
#define LBRACKET        '['
#define NOT             '!'
#define QUESTION        '?'
#define QUOTE           '\\'
#define RANGE           '-'
#define RBRACKET        ']'
#define SEP             '/'
#define STAR            '*'
#define TILDE           '~'
#define UNDERSCORE      '_'
#define LBRACE          '{'
#define RBRACE          '}'
#define SLASH           '/'
#define COMMA           ','
#define COLON           ':'
#define INITIAL 8               /* initial pathv allocation */
#define NULLCPP ((char **)0)    /* Null char ** */
#define NAME_MAX        1024    /* something large */

#define M_QUOTE         0x800000
#define M_PROTECT       0x400000
static int      globit(size_t, const char *, glob_t *, int,
         int (*)(const char *, int), char **);
static int      pstrcmp(const void *, const void *);
static int      append(glob_t *, const char *);

typedef struct Char_s {
        wchar_t wc;
        uint_t at;
} Char;

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

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

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

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

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

static int       compare(const void *, const void *);
static int       compare_gps(const void *, const void *);
static int       g_Ctoc(const Char *, char *, uint_t);
static int       g_lstat(Char *, struct stat *, glob_t *);
static DIR      *g_opendir(Char *, glob_t *);
static Char     *g_strchr(const Char *, wchar_t);
static int       g_stat(Char *, struct stat *, glob_t *);
```



```
static int       glob0(const Char *, glob_t *, struct glob_lim *,
                       int (*)(const char *, int));
static int       glob1(Char *, Char *, glob_t *, struct glob_lim *,
                       int (*)(const char *, int));
static int       glob2(Char *, Char *, Char *, Char *, Char *, Char *,
                       glob_t *, struct glob_lim *,
                       int (*)(const char *, int));
static int       glob3(Char *, Char *, Char *, Char *, Char *,
                       Char *, Char *, glob_t *, struct glob_lim *,
                       int (*)(const char *, int));
static int       globextend(const Char *, glob_t *, struct glob_lim *,
                   struct stat *);
static
const Char      *globtilde(const Char *, Char *, size_t, glob_t *);
static int       globexp1(const Char *, glob_t *, struct glob_lim *,
                      int (*)(const char *, int));
static int       globexp2(const Char *, const Char *, glob_t *,
                       struct glob_lim *, int (*)(const char *, int));
static int       match(Char *, Char *, Char *, int);
#ifdef DEBUG
static void      qprintf(const char *, Char *);
#endif

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

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

        patnext = pattern;
        if (!(flags & GLOB_APPEND)) {
                pglob->gl_pathc = 0;
                pglob->gl_pathv = NULL;
                if ((flags & GLOB_KEEPSTAT) != 0)
                        pglob->gl_statv = NULL;
                if (!(flags & GLOB_DOOFFS))
                        pglob->gl_offs = 0;
        }
        pglob->gl_flags = flags & ~GLOB_MAGCHAR;
        pglob->gl_matchc = 0;

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

        bufnext = patbuf;
        bufend = bufnext + MAXPATHLEN - 1;
        if (flags & GLOB_NOESCAPE) {
                while (bufnext < bufend) {
                        if ((n = mbtowc(&c, patnext, PATH_MAX)) > 0) {
                                patnext += n;
                                bufnext->at = 0;
                                (bufnext++)->wc = c;
                        } else if (n == 0) {
                                break;
                        } else {
                                return (GLOB_NOMATCH);
                        }
```

```
228                 }
229         } else {
230                 /* Protect the quoted characters. */
231                 while (bufnext < bufend) {
232                         if ((n = mbtowc(&c, patnext, PATH_MAX)) > 0) {
233                                 patnext += n;
234                                 if (c == QUOTE) {
235                                         n = mbtowc(&c, patnext, PATH_MAX);
236                                         if (n < 0)
237                                                 return (GLOB_NOMATCH);
238                                         if (n > 0)
239                                                 patnext += n;
240                                         if (n == 0)
241                                                 c = QUOTE;
242                                         bufnext->at = M_PROTECT;
243                                         (bufnext++)->wc = c;
244                                 } else {
245                                         bufnext->at = 0;
246                                         (bufnext++)->wc = c;
247                                 }
248                         } else if (n == 0) {
249                                 break;
250                         } else {
251                                 return (GLOB_NOMATCH);
252                         }
253                 }
254         }
255         bufnext->at = 0;
256         bufnext->wc = EOS;

258         if (flags & GLOB_BRACE)
259                 return (globexp1(patbuf, pglob, &limit, errfunc));
260         else
261                 return (glob0(patbuf, pglob, &limit, errfunc));
262 }

264 /*
265  * Expand recursively a glob {} pattern. When there is no more expansion
266  * invoke the standard globbing routine to glob the rest of the magic
267  * characters
 75  * Free all space consumed by glob.
268  */
269 static int
270 globexp1(const Char *pattern, glob_t *pglob, struct glob_lim *limitp,
271     int (*errfunc)(const char *, int))
 77 void
 78 globfree(glob_t *gp)
272 {
273         const Char* ptr = pattern;
 80         size_t i;

275         /* Protect a single {}, for find(1), like csh */
276         if (pattern[0].wc == LBRACE && pattern[1].wc == RBRACE &&
277             pattern[2].wc == EOS)
278                 return (glob0(pattern, pglob, limitp, errfunc));
 82         if (gp->gl_pathv == 0)
 83                 return;

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

283         return (glob0(pattern, pglob, limitp, errfunc));
 89         gp->gl_pathc = 0;
```



```
 90         gp->gl_pathv = NULLCPP;
284 }


287 /*
288  * Recursive brace globbing helper. Tries to expand a single brace.
289  * If it succeeds then it invokes globexp1 with the new pattern.
290  * If it fails then it tries to glob the rest of the pattern and returns.
 94  * Do filename expansion.
291  */
292 static int
293 globexp2(const Char *ptr, const Char *pattern, glob_t *pglob,
294     struct glob_lim *limitp, int (*errfunc)(const char *, int))
 96 int
 97 glob(const char *pattern, int flags,
 98         int (*errfn)(const char *, int), glob_t *gp)
295 {
296         int     i, rv;
297         Char   *lm, *ls;
298         const Char *pe, *pm, *pl;
299         Char    patbuf[MAXPATHLEN];
100         int rv;
101         size_t i;
102         size_t ipathc;
103         char    *path;

301         /* copy part up to the brace */
302         for (lm = patbuf, pm = pattern; pm != ptr; *lm++ = *pm++)
303                 ;
304         lm->at = 0;
305         lm->wc = EOS;
306         ls = lm;
105         if ((flags & GLOB_DOOFFS) == 0)
106                 gp->gl_offs = 0;

308         /* Find the balanced brace */
309         for (i = 0, pe = ++ptr; pe->wc != EOS; pe++)
310                 if (pe->wc == LBRACKET) {
311                         /* Ignore everything between [] */
312                         for (pm = pe++; pe->wc != RBRACKET &&
313                             pe->wc != EOS; pe++)
314                                 ;
315                         if (pe->wc == EOS) {
316                                 /*
317                                  * We could not find a matching RBRACKET.
318                                  * Ignore and just look for RBRACE
319                                  */
320                                 pe = pm;
321                         }
322                 } else if (pe->wc == LBRACE) {
323                         i++;
324                 } else if (pe->wc == RBRACE) {
325                         if (i == 0)
326                                 break;
327                         i--;
328                 }
108         if (!(flags & GLOB_APPEND)) {
109                 gp->gl_pathc = 0;
110                 gp->gl_pathn = gp->gl_offs + INITIAL;
111                 gp->gl_pathv = (char **)malloc(sizeof (char *) * gp->gl_pathn);

330         /* Non matching braces; just glob the pattern */
331         if (i != 0 || pe->wc == EOS)
332                 return (glob0(patbuf, pglob, limitp, errfunc));
113                 if (gp->gl_pathv == NULLCPP)
114                         return (GLOB_NOSPACE);
```

```
115                 gp->gl_pathp = gp->gl_pathv + gp->gl_offs;

334         for (i = 0, pl = pm = ptr; pm <= pe; pm++) {
335                 switch (pm->wc) {
336                 case LBRACKET:
337                         /* Ignore everything between [] */
338                         for (pl = pm++; pm->wc != RBRACKET && pm->wc != EOS;
339                             pm++)
340                                 ;
341                         if (pm->wc == EOS) {
342                                 /*
343                                  * We could not find a matching RBRACKET.
344                                  * Ignore and just look for RBRACE
345                                  */
346                                 pm = pl;
117                 for (i = 0; i < gp->gl_offs; ++i)
118                         gp->gl_pathv[i] = NULL;
347                         }
348                         break;

350                 case LBRACE:
351                         i++;
352                         break;
121         if ((path = malloc(strlen(pattern)+1)) == NULL)
122                 return (GLOB_NOSPACE);

354                 case RBRACE:
355                         if (i) {
356                                 i--;
357                                 break;
358                         }
359                         /* FALLTHROUGH */
360                 case COMMA:
361                         if (i && pm->wc == COMMA)
362                                 break;
363                         else {
364                                 /* Append the current string */
365                                 for (lm = ls; (pl < pm); *lm++ = *pl++)
366                                         ;
124         ipathc = gp->gl_pathc;
125         rv = globit(0, pattern, gp, flags, errfn, &path);

127         if (rv == GLOB_ABORTED) {
368                                 /*
369                                  * Append the rest of the pattern after the
370                                  * closing brace
129                  * User's error function returned non-zero, or GLOB_ERR was
130                  * set, and we encountered a directory we couldn't search.
371                                  */
372                                 for (pl = pe + 1;
373                                     (*lm++ = *pl++).wc != EOS; /* */)
374                                         ;

376                                 /* Expand the current pattern */
377                                 rv = globexp1(patbuf, pglob, limitp, errfunc);
378                                 if (rv && rv != GLOB_NOMATCH)
379                                         return (rv);

381                                 /* move after the comma, to the next string */
382                                 pl = pm + 1;
132                 free(path);
133                 return (GLOB_ABORTED);
383                         }
384                         break;

386                 default:
```



```
387                         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);
388                 }
141         if (i == 0) {
142                 if (flags & GLOB_NOCHECK)
143                         (void) append(gp, pattern);
144                 else
145                         rv = GLOB_NOMATCH;
389         }
390         return (0);
147         gp->gl_pathp[gp->gl_pathc] = NULL;
148         free(path);

150         return (rv);
391 }



395 /*
396  * expand tilde from the passwd file.
155  * Recursive routine to match glob pattern, and walk directories.
397  */
398 static const Char *
399 globtilde(const Char *pattern, Char *patbuf, size_t patbuf_len, glob_t *pglob)
157 int
158 globit(size_t dend, const char *sp, glob_t *gp, int flags,
159         int (*errfn)(const char *, int), char **path)
400 {
401         struct passwd *pwd;
402         char *h;
403         const Char *p;
404         Char *b, *eb, *q;
405         size_t n;
406         wchar_t c;
162         size_t m;
163         ssize_t end = 0;        /* end of expanded directory */
164         char *pat = (char *)sp; /* pattern component */
165         char *dp = (*path) + dend;
166         int expand = 0;         /* path has pattern */
167         char *cp;
168         struct stat64 sb;
169         DIR *dirp;
170         struct dirent64 *d;
171         int err;

408         if (pattern->wc != TILDE || !(pglob->gl_flags & GLOB_TILDE))
409                 return (pattern);

411         /* Copy up to the end of the string or / */
412         eb = &patbuf[patbuf_len - 1];
413         for (p = pattern + 1, q = patbuf;
414             q < eb && p->wc != EOS && p->wc != SLASH; *q++ = *p++)
415                 ;

417         q->at = 0;
418         q->wc = EOS;

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

422         if (patbuf[0].wc == EOS) {
423                 /*
424                  * handle a plain ~ or ~/ by expanding $HOME
425                  * first and then trying the password file
```

```
426                  */
427                 if (issetugid() != 0 || (h = getenv("HOME")) == NULL) {
428                         if ((pwd = getpwuid(getuid())) == NULL)
429                                 return (pattern);
430                         else
431                                 h = pwd->pw_dir;
173         for (;;)
174                 switch (*dp++ = *(unsigned char *)sp++) {
175                 case '\0':      /* end of source path */
176                         if (expand)
177                                 goto Expand;
178                         else {
179                                 if (!(flags & GLOB_NOCHECK) ||
180                                     flags & (GLOB__CHECK|GLOB_MARK))
181                                         if (stat64(*path, &sb) < 0) {
182                                                 return (0);
432                 }
433         } else {
434                 /*
435                  * Expand a ~user
436                  */
437                 if ((pwd = getpwnam((char *)patbuf)) == NULL)
438                         return (pattern);
439                 else
440                         h = pwd->pw_dir;
184                         if (flags & GLOB_MARK && S_ISDIR(sb.st_mode)) {
185                                 *dp = '\0';
186                                 *--dp = '/';
441         }

443         /* Copy the home directory */
444         for (b = patbuf; b < eb && *h != EOS; b++) {
445                 if ((n = mbtowc(&c, h, PATH_MAX)) > 0) {
446                         h += n;
447                         b->at = 0;
448                         b->wc = c;
449                 } else {
450                         break;
188                         if (append(gp, *path) < 0) {
189                                 return (GLOB_NOSPACE);
451                 }
191                         return (0);
452         }
193                         /*NOTREACHED*/

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

460         return (patbuf);
461 }

463 static int
464 g_charclass(const Char **patternp, Char **bufnextp)
465 {
466         const Char *pattern = *patternp + 1;
467         Char *bufnext = *bufnextp;
468         const Char *colon;
469         struct cclass *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         for (cc = cclasses; cc->name != NULL; cc++) {
478                 wchar_t w;
479                 const Char *s1 = pattern;
480                 const char *s2 = cc->name;
481                 size_t n = len;

483                 /* Are the strings the same? */
484                 while (n > 0 && (w = (s1++)->wc) == *s2 && w != EOS) {
485                         n--;
486                         s2++;
487                 }
488                 if (n == 0 && *s2 == EOS)
195                 case '*':
196                 case '?':
197                 case '[':
198                 case '\\':
199                         ++expand;
489                         break;
490         }
491         if (cc->name == NULL)
492                 return (-1);    /* invalid character class */
493         bufnext->at = M_QUOTE;
494         (bufnext++)->wc = M_CLASS;
495         bufnext->at = (cc - &cclasses[0]);
496         (bufnext++)->wc = COLON;
497         *bufnextp = bufnext;
498         *patternp += len + 3;

500         return (0);
501 }

503 /*
504  * The main glob() routine: compiles the pattern (optionally processing
505  * quotes), calls glob1() to do the real pattern matching, and finally
506  * sorts the list (unless unsorted operation is requested).  Returns 0
507  * if things went well, nonzero if errors occurred.  It is not an error
508  * to find no matches.
509  */
510 static int
511 glob0(const Char *pattern, glob_t *pglob, struct glob_lim *limitp,
512     int (*errfunc)(const char *, int))
513 {
514         const Char *qpatnext;
515         int err, oldpathc;
516         wchar_t c;
517         int a;
518         Char *bufnext, patbuf[MAXPATHLEN];

520         qpatnext = globtilde(pattern, patbuf, MAXPATHLEN, pglob);
521         oldpathc = pglob->gl_pathc;
522         bufnext = patbuf;

524         /* We don't need to check for buffer overflow any more. */
525         while ((a = qpatnext->at), (c = (qpatnext++)->wc) != EOS) {
526                 switch (c) {
527                 case LBRACKET:
528                         if (a != 0) {
529                                 bufnext->at = a;
530                                 (bufnext++)->wc = c;
202                 case '/':
203                         if (expand)
204                                 goto Expand;
205                         end = dp - *path;
206                         pat = (char *)sp;
```

```
531                             break;
532                     }
533                     a = qpatnext->at;
534                     c = qpatnext->wc;
535                     if (a == 0 && c == NOT)
536                             ++qpatnext;
537                     if (qpatnext->wc == EOS ||
538                         g_strchr(qpatnext+1, RBRACKET) == NULL) {
539                             bufnext->at = 0;
540                             (bufnext++)->wc = LBRACKET;
541                             if (a == 0 && c == NOT)
542                                     --qpatnext;
543                             break;
544                     }
545                     bufnext->at = M_QUOTE;
546                     (bufnext++)->wc = M_SET;
547                     if (a == 0 && c == NOT) {
548                             bufnext->at = M_QUOTE;
549                             (bufnext++)->wc = M_NOT;
550                     }
551                     a = qpatnext->at;
552                     c = (qpatnext++)->wc;
553                     do {
554                             if (a == 0 && c == LBRACKET &&
555                                 qpatnext->wc == COLON) {
556                                     do {
557                                             err = g_charclass(&qpatnext,
558                                                 &bufnext);
559                                             if (err)
560                                                     break;
561                                             a = qpatnext->at;
562                                             c = (qpatnext++)->wc;
563                                     } while (a == 0 && c == LBRACKET &&
564                                         qpatnext->wc == COLON);
565                                     if (err == -1 &&
566                                         !(pglob->gl_flags & GLOB_NOCHECK))
567                                             return (GLOB_NOMATCH);
568                                     if (a == 0 && c == RBRACKET)
569                                             break;
570                             }
571                             bufnext->at = a;
572                             (bufnext++)->wc = c;
573                             if (qpatnext->at == 0 &&
574                                 qpatnext->wc == RANGE) {
575                                     a = qpatnext[1].at;
576                                     c = qpatnext[1].wc;
577                                     if (qpatnext[1].at != 0 ||
578                                         qpatnext[1].wc != RBRACKET) {
579                                             bufnext->at = M_QUOTE;
580                                             (bufnext++)->wc = M_RNG;
581                                             bufnext->at = a;
582                                             (bufnext++)->wc = c;
583                                             qpatnext += 2;
584                                     }
585                             }
586                             a = qpatnext->at;
587                             c = (qpatnext++)->wc;
588                     } while (a != 0 || c != RBRACKET);
589                     pglob->gl_flags |= GLOB_MAGCHAR;
590                     bufnext->at = M_QUOTE;
591                     (bufnext++)->wc = M_END;
592                     break;
593             case QUESTION:
594                     if (a != 0) {
595                             bufnext->at = a;
596                             (bufnext++)->wc = c;
```



```
597                             break;
598                     }
599                     pglob->gl_flags |= GLOB_MAGCHAR;
600                     bufnext->at = M_QUOTE;
601                     (bufnext++)->wc = M_ONE;
602                     break;
603             case STAR:
604                     if (a != 0) {
605                             bufnext->at = a;
606                             (bufnext++)->wc = c;
607                             break;
608                     }
609                     pglob->gl_flags |= GLOB_MAGCHAR;
610                     /*
611                      * collapse adjacent stars to one,
612                      * to avoid exponential behavior
613                      */
614                     if (bufnext == patbuf ||
615                         bufnext[-1].at != M_QUOTE ||
616                         bufnext[-1].wc != M_ALL) {
617                             bufnext->at = M_QUOTE;
618                             (bufnext++)->wc = M_ALL;
619                     }
620                     break;
621             default:
622                     bufnext->at = a;
623                     (bufnext++)->wc = c;
624                     break;
625             }
626     }
627     bufnext->at = 0;
628     bufnext->wc = EOS;
629 #ifdef DEBUG
630     qprintf("glob0:glob1:patbuf", patbuf);
631 #endif

633     if ((err = glob1(patbuf, patbuf+MAXPATHLEN-1, pglob, limitp, errfunc))
634         != 0)
635             return (err);

637     /*
638      * If there was no match we are going to append the pattern
639      * if GLOB_NOCHECK was specified or if GLOB_NOMAGIC was specified
640      * and the pattern did not contain any magic characters
641      * GLOB_NOMAGIC is there just for compatibility with csh.
642      */
643     if (pglob->gl_pathc == oldpathc) {
644             if ((pglob->gl_flags & GLOB_NOCHECK) ||
645                 ((pglob->gl_flags & GLOB_NOMAGIC) &&
646                  !(pglob->gl_flags & GLOB_MAGCHAR)))
647                     return (globextend(pattern, pglob, limitp, NULL));
648             else
649                     return (GLOB_NOMATCH);
209             Expand:
210                     /* determine directory and open it */
211                     (*path)[end] = '\0';
212                     dirp = opendir(**path == '\0' ? "." : *path);
213                     if (dirp == NULL) {
214                             if (errfn != 0 && errfn(*path, errno) != 0 ||
215                                 flags&GLOB_ERR) {
216                                     return (GLOB_ABORTED);
650     }
651     if (!(pglob->gl_flags & GLOB_NOSORT)) {
652             if ((pglob->gl_flags & GLOB_KEEPSTAT)) {
653                     /* Keep the paths and stat info synced during sort */
654                     struct glob_path_stat *path_stat;
```

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

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

681 static int
682 compare(const void *p, const void *q)
683 {
684         return (strcmp(*(char **)p, *(char **)q));
685 }

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

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

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

702         /* A null pathname is invalid -- POSIX 1003.1 sect. 2.4. */
703         if (pattern->wc == EOS)
704                 return (0);
705         return (glob2(pathbuf, pathbuf+MAXPATHLEN-1,
706             pathbuf, pathbuf+MAXPATHLEN-1,
707             pattern, pattern_last, pglob, limitp, errfunc));
708 }

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



```
721         Char *p, *q;
722         int anymeta;

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

733                         if ((pglob->gl_flags & GLOB_LIMIT) &&
734                             limitp->glim_stat++ >= GLOB_LIMIT_STAT) {
735                                 errno = 0;
736                                 pathend->at = 0;
737                                 (pathend++)->wc = SEP;
738                                 pathend->at = 0;
739                                 pathend->wc = EOS;
740                                 return (GLOB_NOSPACE);
741                         }
742                         if (g_lstat(pathbuf, &sb, pglob))
743                                 return (0);

745                         if (((pglob->gl_flags & GLOB_MARK) &&
746                             (pathend[-1].at != 0 ||
747                             pathend[-1].wc != SEP)) &&
748                             (S_ISDIR(sb.st_mode) ||
749                             (S_ISLNK(sb.st_mode) &&
750                             (g_stat(pathbuf, &sb, pglob) == 0) &&
751                             S_ISDIR(sb.st_mode)))) {
752                                 if (pathend+1 > pathend_last)
221                         /* extract pattern component */
222                         n = sp - pat;
223                         if ((cp = malloc(n)) == NULL) {
224                                 (void) closedir(dirp);
753                                         return (GLOB_NOSPACE);
754                                 pathend->at = 0;
755                                 (pathend++)->wc = SEP;
756                                 pathend->at = 0;
757                                 pathend->wc = EOS;
758                         }
759                         ++pglob->gl_matchc;
760                         return (globextend(pathbuf, pglob, limitp, &sb));
761                 }
227                         pat = memcpy(cp, pat, n);
228                         pat[n-1] = '\0';
229                         if (*--sp != '\0')
230                                 flags |= GLOB__CHECK;

763                 /* Find end of next segment, copy tentatively to pathend. */
764                 q = pathend;
765                 p = pattern;
766                 while (p->wc != EOS && p->wc != SEP) {
767                         if (ismeta(*p))
768                                 anymeta = 1;
769                         if (q+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);
770                                 return (GLOB_NOSPACE);
771                         *q++ = *p++;
```

```
772                 }
241                         dp = (*path) + n;

774                 if (!anymeta) {         /* No expansion, do next segment. */
775                         pathend = q;
776                         pattern = p;
777                         while (pattern->wc == SEP) {
778                                 if (pathend+1 > pathend_last)
779                                         return (GLOB_NOSPACE);
780                                 *pathend++ = *pattern++;
781                         }
782                 } else  {
783                         /* Need expansion, recurse. */
784                         return (glob3(pathbuf, pathbuf_last, pathend,
785                             pathend_last, pattern, p, pattern_last,
786                             pglob, limitp, errfunc));
787                 }
788         }
789         /* NOTREACHED */
790 }

792 static int
793 glob3(Char *pathbuf, Char *pathbuf_last, Char *pathend, Char *pathend_last,
794     Char *pattern, Char *restpattern, Char *restpattern_last, glob_t *pglob,
795     struct glob_lim *limitp, int (*errfunc)(const char *, int))
796 {
797         struct dirent *dp;
798         DIR *dirp;
799         int err;
800         char buf[MAXPATHLEN];

802         /*
803          * The readdirfunc declaration can't be prototyped, because it is
804          * assigned, below, to two functions which are prototyped in glob.h
805          * and dirent.h as taking pointers to differently typed opaque
806          * structures.
807          */
808         struct dirent *(*readdirfunc)(void *);

810         if (pathend > pathend_last)
811                 return (GLOB_NOSPACE);
812         pathend->at = 0;
813         pathend->wc = EOS;
814         errno = 0;

816         if ((dirp = g_opendir(pathbuf, pglob)) == NULL) {
817                 /* TODO: don't call for ENOENT or ENOTDIR? */
818                 if (errfunc) {
819                         if (g_Ctoc(pathbuf, buf, sizeof (buf)))
820                                 return (GLOB_ABORTED);
821                         if (errfunc(buf, errno) ||
822                             pglob->gl_flags & GLOB_ERR)
823                                 return (GLOB_ABORTED);
824                 }
825                 return (0);
826         }

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

830         /* Search directory for matching names. */
831         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
832                 readdirfunc = pglob->gl_readdir;
833         else
834                 readdirfunc = (struct dirent *(*)(void *))readdir;
835         while ((dp = (*readdirfunc)(dirp))) {
```



```
836                 char *sc;
837                 Char *dc;
838                 size_t n;
839                 wchar_t w;

841                 if ((pglob->gl_flags & GLOB_LIMIT) &&
842                     limitp->glim_readdir++ >= GLOB_LIMIT_READDIR) {
843                         errno = 0;
844                         pathend->at = 0;
845                         (pathend++)->wc = SEP;
846                         pathend->at = 0;
847                         pathend->wc = EOS;
848                         err = GLOB_NOSPACE;
849                         break;
850                 }

852                 /* Initial DOT must be matched literally. */
853                 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))
854                         continue;
855                 dc = pathend;
856                 sc = dp->d_name;
857                 while (dc < pathend_last) {
858                         if ((n = mbtowc(&w, sc, MB_LEN_MAX)) <= 0) {
859                                 sc += 1;
860                                 dc->at = 0;
861                                 dc->wc = EOS;
862                         } else {
863                                 sc += n;
864                                 dc->at = 0;
865                                 dc->wc = w;
866                         }
867                         dc++;
868                         if (n <= 0)
869                                 break;
870                 }
871                 if (dc >= pathend_last) {
872                         dc->at = 0;
873                         dc->wc = EOS;
874                         err = GLOB_NOSPACE;
875                         break;
876                 }
877                 if (n < 0) {
878                         dc->at = 0;
879                         dc->wc = EOS;
880                         continue;
881                 }

883                 if (!match(pathend, pattern, restpattern, GLOB_LIMIT_RECUR)) {
884                         pathend->at = 0;
885                         pathend->wc = EOS;
886                         continue;
887                 }
888                 err = glob2(pathbuf, pathbuf_last, --dc, pathend_last,
889                     restpattern, restpattern_last, pglob, limitp,
890                     errfunc);
891                 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)
892                         break;
893         }

895         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
896                 (*pglob->gl_closedir)(dirp);
897         else
898                 closedir(dirp);
261                         (void) closedir(dirp);
262                         free(pat);
899         return (err);
264                 }
265                 /* NOTREACHED */
900 }


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

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



```
950                         free(pglob->gl_statv);
951                         pglob->gl_statv = NULL;
952                 }
953                 return (GLOB_NOSPACE);
954         }

956         pathv = realloc(pglob->gl_pathv, newn * sizeof (*pathv));
957         if (pathv == NULL)
958                 goto nospace;
959         if (pglob->gl_pathv == NULL && pglob->gl_offs > 0) {
960                 /* first time around -- clear initial gl_offs items */
961                 pathv += pglob->gl_offs;
962                 for (i = pglob->gl_offs; --i >= 0; )
963                         *--pathv = NULL;
964         }
965         pglob->gl_pathv = pathv;

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

996         len = MB_LEN_MAX;
997         p = path;
998         while ((n = wctomb(junk, p->wc)) > 0) {
999                 len += n;
1000                if ((p++)->wc == EOS)
1001                        break;
1002        }

1004        limitp->glim_malloc += len;
1005        if ((copy = malloc(len)) != NULL) {
1006                if (g_Ctoc(path, copy, len)) {
1007                        free(copy);
1008                        return (GLOB_NOSPACE);
1009                }
1010                pathv[pglob->gl_offs + pglob->gl_pathc++] = copy;
1011        }
1012        pathv[pglob->gl_offs + pglob->gl_pathc] = NULL;

1014        if ((pglob->gl_flags & GLOB_LIMIT) &&
1015            (newn * sizeof (*pathv)) + limitp->glim_malloc >
```

```
            GLOB_LIMIT_MALLOC) {
                errno = 0;
                return (GLOB_NOSPACE);
        }
        copy_error:
        return (copy == NULL ? GLOB_NOSPACE : 0);
        return (strcoll(*(char **)npp1, *(char **)npp2));
}


/*
 * pattern matching function for filenames.  Each occurrence of the *
 * pattern causes a recursion level.
 * Add a new matched filename to the glob_t structure, increasing the
 * size of that array, as required.
 */
static int
match(Char *name, Char *pat, Char *patend, int recur)
int
append(glob_t *gp, const char *str)
{
        int ok, negate_range;
        Char c, k;
        char *cp;

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

        while (pat < patend) {
                c = *pat++;
                switch (c.wc) {
                case M_ALL:
                        if (c.at != M_QUOTE) {
                                k = *name++;
                                if (k.at != c.at || k.wc != c.wc)
                                        return (0);
                                break;
        if ((gp->gl_pathc + gp->gl_offs) >= gp->gl_pathn) {
                gp->gl_pathn *= 2;
                gp->gl_pathv = (char **)realloc((void *)gp->gl_pathv,
                    gp->gl_pathn * sizeof (char *));
                if (gp->gl_pathv == NULLCPP)
                        return (GLOB_NOSPACE);
                gp->gl_pathp = gp->gl_pathv + gp->gl_offs;
                        }
                        while (pat < patend && pat->at == M_QUOTE &&
                            pat->wc == M_ALL)
                                pat++;  /* eat consecutive '*' */
                        if (pat == patend)
                                return (1);
                        do {
                                if (match(name, pat, patend, recur))
                                        return (1);
                        } while ((name++)->wc != EOS);
                        return (0);
                case M_ONE:
                        if (c.at != M_QUOTE) {
                                k = *name++;
                                if (k.at != c.at || k.wc != c.wc)
                                        return (0);
                                break;
                        }
                        if ((name++)->wc == EOS)
```



```
                                        return (0);
                                break;
                        case M_SET:
                                if (c.at != M_QUOTE) {
                                        k = *name++;
                                        if (k.at != c.at || k.wc != c.wc)
                                                return (0);
                                        break;
                                }
                                ok = 0;
                                if ((k = *name++).wc == EOS)
                                        return (0);
                                if ((negate_range = (pat->at == M_QUOTE &&
                                    pat->wc == M_NOT)) != 0)
                                        ++pat;
                                while (((c = *pat++).at != M_QUOTE) || c.wc != M_END) {
                                        if (c.at == M_QUOTE && c.wc == M_CLASS) {
                                                Char idx;

                                                idx.at = pat->at;
                                                idx.wc = pat->wc;
                                                if (idx.at < NCCLASSES &&
                                                    cclasses[idx.at].isctype(k.wc))
                                                        ok = 1;
                                                ++pat;
                                        }
                                        if (pat->at == M_QUOTE && pat->wc == M_RNG) {
                                                if (c.wc <= k.wc && k.wc <= pat[1].wc)
                                                        ok = 1;
                                                pat += 2;
                                        } else if (c.wc == k.wc)
                                                ok = 1;
                                }
                                if (ok == negate_range)
                                        return (0);
                                break;
                        default:
                                k = *name++;
                                if (k.at != c.at || k.wc != c.wc)
                                        return (0);
                                break;
                        }
                }
                return (name->wc == EOS);
}

/* Free allocated data belonging to a glob_t structure. */
void
globfree(glob_t *pglob)
{
        int i;
        char **pp;

        if (pglob->gl_pathv != NULL) {
                pp = pglob->gl_pathv + pglob->gl_offs;
                for (i = pglob->gl_pathc; i--; ++pp)
                        if (*pp)
                                free(*pp);
                free(pglob->gl_pathv);
                pglob->gl_pathv = NULL;
        }
        if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
            pglob->gl_statv != NULL) {
                for (i = 0; i < pglob->gl_pathc; i++) {
                        if (pglob->gl_statv[i] != NULL)
                                free(pglob->gl_statv[i]);
```

```
1132                 }
1133                 free(pglob->gl_statv);
1134                 pglob->gl_statv = NULL;
1135         }
1136 }

1138 static DIR *
1139 g_opendir(Char *str, glob_t *pglob)
1140 {
1141         char buf[MAXPATHLEN];

1143         if (str->wc == EOS)
1144                 strlcpy(buf, ".", sizeof (buf));
1145         else {
1146                 if (g_Ctoc(str, buf, sizeof (buf)))
1147                         return (NULL);
1148         }

1150         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1151                 return ((*pglob->gl_opendir)(buf));

1153         return (opendir(buf));
1154 }

1156 static int
1157 g_lstat(Char *fn, struct stat *sb, glob_t *pglob)
1158 {
1159         char buf[MAXPATHLEN];

1161         if (g_Ctoc(fn, buf, sizeof (buf)))
1162                 return (-1);
1163         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1164                 return ((*pglob->gl_lstat)(buf, sb));
1165         return (lstat(buf, sb));
1166 }

1168 static int
1169 g_stat(Char *fn, struct stat *sb, glob_t *pglob)
1170 {
1171         char buf[MAXPATHLEN];

1173         if (g_Ctoc(fn, buf, sizeof (buf)))
1174                 return (-1);
1175         if (pglob->gl_flags & GLOB_ALTDIRFUNC)
1176                 return ((*pglob->gl_stat)(buf, sb));
1177         return (stat(buf, sb));
1178 }

1180 static Char *
1181 g_strchr(const Char *str, wchar_t ch)
1182 {
1183         do {
1184                 if (str->at == 0 && str->wc == ch)
1185                         return ((Char *)str);
1186         } while ((str++)->wc != EOS);
1187         return (NULL);
1188 }

1190 static int
1191 g_Ctoc(const Char *str, char *buf, uint_t len)
1192 {
1193         int n;
1194         wchar_t w;

1196         while (len >= MB_LEN_MAX) {
1197                 w = (str++)->wc;
```



```
1198                 if ((n = wctomb(buf, w)) > 0) {
1199                         len -= n;
1200                         buf += n;
1201                 }
1202                 if (n < 0)
1203                         break;
1204                 if (w == EOS)
1205                         return (0);
1206         }
1207         return (1);
1208 }

1210 #ifdef DEBUG
1211 static void
1212 qprintf(const char *str, Char *s)
1213 {
1214         Char *p;

1216         (void) printf("%s:\n", str);
1217         for (p = s; p->wc != EOS; p++)
1218                 (void) printf("%wc", p->wc);
1219         (void) printf("\n");
1220         for (p = s; p->wc != EOS; p++)
1221                 (void) printf("%c", p->at & M_PROTECT ? '"' : ' ');
1222         (void) printf("\n");
1223         for (p = s; p->wc != EOS; p++)
1224                 (void) printf("%c", ismeta(*p) ? '_' : ' ');
1225         (void) printf("\n");
1226 }
1227 #endif
```

```
**********************************************************
   17719 Mon Dec 17 10:05:18 2012
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()
**********************************************************
'\" te
.\" Copyright (c) 1992, X/Open Company Limited. All Rights Reserved.
.\" Portions Copyright (c) 2003, Sun Microsystems, Inc.  All Rights Reserved.
.\" Portions Copyright (c) 2012, Gary Mills
.\" Copyright (c) 1992, X/Open Company Limited. All Rights Reserved.  Portions C
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" http://www.opengroup.org/bookstore/.
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\"
.\" $OpenBSD: glob.3,v 1.30 2012/01/20 07:09:42 tedu Exp $
.\"
.\" Copyright (c) 1989, 1991, 1993, 1994
.\" The Regents of the University of California.  All rights reserved.
.\"
.\" This code is derived from software contributed to Berkeley by
.\" Guido van Rossum.
.\" Redistribution and use in source and binary forms, with or without
.\" modification, are permitted provided that the following conditions
.\" are met:
.\" 1. Redistributions of source code must retain the above copyright
.\"    notice, this list of conditions and the following disclaimer.
.\" 2. Redistributions in binary form must reproduce the above copyright
.\"    notice, this list of conditions and the following disclaimer in the
.\"    documentation and/or other materials provided with the distribution.
.\" 3. Neither the name of the University nor the names of its contributors
.\"    may be used to endorse or promote products derived from this software
.\"    without specific prior written permission.
.\"
.\" THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
.\" ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
.\" IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
.\" ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
.\" FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
.\" DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
.\" OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
.\" HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
.\" LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
.\" OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
.\" SUCH DAMAGE.
.\"
.\"  This notice shall appear on any product containing this material.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH GLOB 3C "Nov 1, 2003"
.SH NAME
glob, globfree \- generate path names matching a pattern
.SH SYNOPSIS
.LP
.nf
#include <glob.h>

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

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



```
.fi

.SH DESCRIPTION
.sp
.LP
The \fBglob()\fR function is a path name generator.
.sp
.LP
The \fBglobfree()\fR function frees any memory allocated by \fBglob()\fR
associated with \fIpglob\fR.
.SS "\fIpattern\fR Argument"
.sp
.LP
The argument \fIpattern\fR is a pointer to a path name pattern to be expanded.
The \fBglob()\fR function matches all accessible path names against this
pattern and develops a list of all path names that match. In order to have
access to a path name, \fBglob()\fR requires search permission on every
component of a path except the last, and read permission on each directory of
any filename component of \fIpattern\fR that contains any of the following
special characters:
.sp
.in +2
.nf
*        ?        [
.fi
.in -2

.SS "\fIpglob\fR Argument"
.sp
.LP
The structure type \fBglob_t\fR is defined in the header \fB<glob.h>\fR and
includes at least the following members:
.sp
.in +2
.nf
size_t   gl_pathc;     /* Total count of paths matched by */
size_t   gl_pathc;     /* count of paths matched by */
                       /* pattern */
char     **gl_pathv;   /* List of matched path names */
size_t   gl_offs;      /* # of slots reserved in gl_pathv */
int      gl_matchc;    /* Count of paths matching pattern. */
int      gl_flags;     /* Copy of flags parameter to glob. */
char     **gl_pathv;   /* pointer to list of matched */
                       /* path names */
size_t   gl_offs;      /* slots to reserve at beginning */
                       /* of gl_pathv */
.fi
.in -2

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

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

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

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

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

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

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



```
187 .RE

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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

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



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

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

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

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

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

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

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

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



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

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

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

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

.sp
.LP
Using the same example:

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

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

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

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
```

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

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