```
**********************************************************
   26964 Thu Oct 10 15:32:54 2013
new/usr/src/cmd/ssh/include/config.h
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
/* config.h.  Generated by configure.  */
/* config.h.in.  Generated from configure.ac by autoheader.  */
/* $Id: acconfig.h,v 1.145 2002/09/26 00:38:48 tim Exp $ */

/*
 * Copyright (c) 2001, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2013 Gary Mills
 */

#ifndef _CONFIG_H
#define _CONFIG_H

#ifdef __cplusplus
extern "C" {
#endif


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


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

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

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

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

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

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

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

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

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

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

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

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

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



```
#define IP_TOS_IS_BROKEN 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
127 #define ALTPRIVSEP

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

388 /*
```

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

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

 397 /* Define on *nto-qnx systems */
 398 /* #undef MISSING_NFDBITS */

 400 /* Define on *nto-qnx systems */
 401 /* #undef MISSING_HOWMANY */

 403 /* Define on *nto-qnx systems */
 404 /* #undef MISSING_FD_MASK */

 406 /*
 407  * Use libedit or libtecla for sftp
 408  * If both USE_LIBEDIT and USE_LIBTECLA are defined, then USE_LIBEDIT will
 409  * have higher precedence.
 410  */
 411 #undef USE_LIBEDIT
 412 #define USE_LIBTECLA    1

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

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

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

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

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


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

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

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

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

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

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

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

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

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



```
 455 #define HAVE_CRYPT_H 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
#define HAVE_GETRUSAGE 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



```
#define HAVE_LIBDL 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
653 #define HAVE_NETDB_H 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



```
719 #define HAVE_SETENV 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
#define HAVE_STDLIB_H 1

/* Define to 1 if you have the ‘strerror’ function. */
#define HAVE_STRERROR 1

/* Define to 1 if you have the ‘strftime’ function. */
#define HAVE_STRFTIME 1

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

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

/* Define to 1 if you have the ‘strlcat’ function. */
#define HAVE_STRLCAT 1

/* Define to 1 if you have the ‘strlcpy’ function. */
#define HAVE_STRLCPY 1

/* Define to 1 if you have the ‘strmode’ function. */
/* #undef HAVE_STRMODE */

/* Define to 1 if ‘st_blksize’ is member of ‘struct stat’. */
#define HAVE_STRUCT_STAT_ST_BLKSIZE 1

/* Define to 1 if you have the ‘sysconf’ function. */
#define HAVE_SYSCONF 1

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

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

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


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

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

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

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

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

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

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

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

/* Define to 1 if you have the ‘tcgetpgrp’ function. */
#define HAVE_TCGETPGRP 1
```



```
/* Define to 1 if you have the ‘time’ function. */
#define HAVE_TIME 1

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

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

/* Define to 1 if you have the ‘truncate’ function. */
#define HAVE_TRUNCATE 1

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

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

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

/* Define to 1 if you have the ‘updwtmp’ function. */
#define HAVE_UPDWTMP 1

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

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

/* Define to 1 if you have the ‘utimes’ function. */
#define HAVE_UTIMES 1

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

/* Define to 1 if you have the ‘utmpname’ function. */
#define HAVE_UTMPNAME 1

/* Define to 1 if you have the ‘utmpxname’ function. */
#define HAVE_UTMPXNAME 1

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

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

/* Define to 1 if you have the ‘vasprintf’ function. */
#define HAVE_VASPRINTF 1

/* Define to 1 if you have the ‘vhangup’ function. */
#define HAVE_VHANGUP 1

/* Define to 1 if you have the ‘vsnprintf’ function. */
#define HAVE_VSNPRINTF 1

/* Define to 1 if you have the ‘waitpid’ function. */
#define HAVE_WAITPID 1

/* Define to 1 if you have the ‘_getpty’ function. */
/* #undef HAVE__GETPTY */

/* Define to 1 if you have the ‘__b64_ntop’ function. */
/* #undef HAVE___B64_NTOP */
```

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

 920 /* Define to the full name of this package. */
 921 #define PACKAGE_NAME ""

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

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

 929 /* Define to the version of this package. */
 930 #define PACKAGE_VERSION ""

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

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

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

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

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

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

 950 /*
 951  * Define to 1 if your processor stores words with the most significant byte
 952  * first (like Motorola and SPARC, unlike Intel and VAX).
 953  */
 954 #define WORDS_BIGENDIAN 1

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

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

 962 /*
 963  * Define as '__inline' if that's what the C compiler calls it, or to nothing if
 964  * it is not supported.
 965  */
 966 /* #undef inline */

 968 /* type to use in place of socklen_t if not defined */
 969 /* #undef socklen_t */

 971 /* Define for BSM auditing (Solaris) support */
 972 #define HAVE_BSM 1

 974 /* Define if compiling in ON */
 975 #define SUNW_SSH 1

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

 979 #ifdef __cplusplus
 980 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    6311 Thu Oct 10 15:32:56 2013
new/usr/src/head/glob.h
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 1985, 1992 by Mortice Kern Systems Inc.  All rights reserved.
 * Copyright 2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

/*
 * Copyright 2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright (c) 2013 Gary Mills
 */

#ifndef _GLOB_H
#define _GLOB_H

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

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

#ifdef  __cplusplus
extern "C" {
#endif

typedef struct  glob_t  {
        /*
         * Members specified 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 */

        /*
         * Internal-use members:
         *
         * NB: The next two members are carried in both the
         * libc backward compatibility wrapper functions and
         * the extended functions.
         */
        /* following are internal to the implementation */
        char    **gl_pathp;     /* gl_pathv + gl_offs */
        int     gl_pathn;       /* # of elements allocated */

        /*
         * Non-POSIX extensions
         *
         * NB: The following members are not carried in
         * the libc backward compatibility wrapper functions.
         */
        int     gl_matchc;      /* Count of paths matching pattern. */
        int     gl_flags;       /* Copy of flags parameter to glob. */
        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 *);
}       glob_t;

/*
```

```
 120  * POSIX "flags" argument to glob function.
  53  * "flags" argument to glob function.
 121  */
 122 #define GLOB_ERR        0x0001          /* Don't continue on directory error */
 123 #define GLOB_MARK       0x0002          /* Mark directories with trailing / */
 124 #define GLOB_NOSORT     0x0004          /* Don't sort pathnames */
 125 #define GLOB_NOCHECK    0x0008          /* Return unquoted arg if no match */
 126 #define GLOB_DOOFFS     0x0010          /* Ignore gl_offs unless set */
 127 #define GLOB_APPEND     0x0020          /* Append to previous glob_t */
 128 #define GLOB_NOESCAPE   0x0040          /* Backslashes do not quote M-chars */

 130 /*
 131  * Non-POSIX "flags" argument to glob function, from OpenBSD.
 132  */
 133 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
 134 #define GLOB_POSIX      0x007F  /* All POSIX flags */
 135 #define GLOB_BRACE      0x0080  /* Expand braces ala csh. */
 136 #define GLOB_MAGCHAR    0x0100  /* Pattern had globbing characters. */
 137 #define GLOB_NOMAGIC    0x0200  /* GLOB_NOCHECK without magic chars (csh). */
 138 #define GLOB_QUOTE      0x0400  /* Quote special chars with \. */
 139 #define GLOB_TILDE      0x0800  /* Expand tilde names from the passwd file. */
 140 #define GLOB_LIMIT      0x2000  /* Limit pattern match output to ARG_MAX */
 141 #define GLOB_KEEPSTAT   0x4000  /* Retain stat data for paths in gl_statv. */
 142 #define GLOB_ALTDIRFUNC 0x8000  /* Use alternately specified directory funcs. */
 143 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */

 145 /*
 146  * Error returns from "glob"
 147  */
 148 #define GLOB_NOSYS      (-4)            /* function not supported (XPG4) */
 149 #define GLOB_NOMATCH    (-3)            /* Pattern does not match */
 150 #define GLOB_NOSPACE    (-2)            /* Not enough memory */
 151 #define GLOB_ABORTED    (-1)            /* GLOB_ERR set or errfunc return!=0 */
 152 #define GLOB_ABEND      GLOB_ABORTED    /* backward compatibility */

 154 #ifdef __PRAGMA_REDEFINE_EXTNAME
 155 #pragma redefine_extname        glob    _glob_ext
 156 #pragma redefine_extname        globfree        _globfree_ext
 157 #else /* __PRAGMA_REDEFINE_EXTNAME */
 158 #define glob    _glob_ext
 159 #define globfree        _globfree_ext
 160 #endif /* __PRAGMA_REDEFINE_EXTNAME */

 162 #if defined(__STDC__)

 164 extern int glob(const char *_RESTRICT_KYWD, int, int(*)(const char *, int),
 165                 glob_t *_RESTRICT_KYWD);
 166 extern void globfree(glob_t *);

 168 #else /* __STDC__ */

  75 #else
 170 extern int glob();
 171 extern void globfree();
  78 #endif

 173 #endif /* __STDC__ */

 175 #ifdef  __cplusplus
 176 }
_____unchanged_portion_omitted_
```

```
**********************************************************
   54481 Thu Oct 10 15:32:56 2013
new/usr/src/lib/libc/port/mapfile-vers
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
```

```
#
# 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) 2006, 2010, Oracle and/or its affiliates. All rights reserved.
#
# Copyright 2010 Nexenta Systems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright (c) 2013 Gary Mills
#

#
# MAPFILE HEADER START
#
# WARNING:  STOP NOW.  DO NOT MODIFY THIS FILE.
# Object versioning must comply with the rules detailed in
#
#       usr/src/lib/README.mapfiles
#
# You should not be making modifications here until you've read the most current
# copy of that file. If you need help, contact a gatekeeper for guidance.
#
# MAPFILE HEADER END
#

$mapfile_version 2

#
# All function names added to this or any other libc mapfile
# must be placed under the 'protected:' designation.
# The 'global:' designation is used *only* for data
# items and for the members of the malloc() family.
#

#
# README README README README README README: how to update this file
#   1) each version of Solaris/OpenSolaris gets a version number.
#      (Actually since Solaris is actually a series of OpenSolaris releases
#       we'll just use OpenSolaris for this exercise.)
#       OpenSolaris 2008.11 gets 1.23
#       OpenSolaris 2009.04 gets 1.24
```



```
#       etc.
#   2) each project integration uses a unique version number.
#       PSARC/2008/123 gets 1.24.1
#       PSARC/2008/456 gets 1.24.2
#       etc.
#


# Mnemonic conditional input identifiers:
#
# - amd64, i386, sparc32, sparcv9: Correspond to ISA subdirectories used to
#       hold per-platform code. Note however that we use 'sparc32' instead of
#       'sparc'. Since '_sparc' is predefined to apply to, all sparc platforms,
#       naming the 32-bit version 'sparc' would be too likely to cause errors.
#
# -     lf64: Defined on platforms that offer the 32-bit largefile APIs
#
$if _ELF32
$add lf64
$endif
$if _sparc && _ELF32
$add sparc32
$endif
$if _sparc && _ELF64
$add sparcv9
$endif
$if _x86 && _ELF32
$add i386
$endif
$if _x86 && _ELF64
$add amd64
$endif

SYMBOL_VERSION ILLUMOS_0.4 {    # Illumos additions
    protected:
        _glob_ext;
        _globfree_ext;
} ILLUMOS_0.3;

SYMBOL_VERSION ILLUMOS_0.3 {    # Illumos additions
    protected:
        assfail3;
} ILLUMOS_0.2;
```

*unchanged_portion_omitted_*

```
**********************************************************
   33054 Thu Oct 10 15:32:57 2013
new/usr/src/lib/libc/port/regex/glob.c
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
/*
 * Copyright (c) 2013 Gary Mills
 */
/*      $OpenBSD: glob.c,v 1.39 2012/01/20 07:09:42 tedu Exp $ */
/*
 * Copyright (c) 1989, 1993
 *      The Regents of the University of California.  All rights reserved.
 * CDDL HEADER START
 *
 * This code is derived from software contributed to Berkeley by
 * Guido van Rossum.
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

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

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



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

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

#include "lint.h"
#pragma ident   "%Z%%M% %I%     %E% SMI"

#include <sys/param.h>
#include <sys/stat.h>
#pragma weak _glob = glob
#pragma weak _globfree = globfree

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

/*
 * This is the legacy glob_t prior to illumos enhancement 1097,
 * used when old programs call the old libc glob functions.
```

```
 83  * (New programs call the _glob_ext, _globfree_ext functions.)
 84  * This struct should be considered "carved in stone".
 85  */
 86 typedef struct  old_glob        {
 87         size_t  gl_pathc;               /* Count of paths matched by pattern */
 88         char    **gl_pathv;             /* List of matched pathnames */
 89         size_t  gl_offs;                /* # of slots reserved in gl_pathv */
 90         /* following are internal to the implementation */
 91         char    **gl_pathp;             /* gl_pathv + gl_offs */
 92         int     gl_pathn;               /* # of elements allocated */
 93 }       old_glob_t;
 63 #define GLOB__CHECK     0x80    /* stat generated paths */

 95 /*
 96  * For old programs, the external names need to be the old names:
 97  * glob() and globfree() .  We've redefined those already to
 98  *  _glob_ext() and _globfree_ext() .  Now redefine old_glob()
 99  * and old_globfree() to glob() and globfree() .
100  */
101 #ifdef __PRAGMA_REDEFINE_EXTNAME
102 #pragma redefine_extname        old_glob        glob
103 #pragma redefine_extname        old_globfree    globfree
104 #endif /* __PRAGMA_REDEFINE_EXTNAME */
105 extern int old_glob(const char *, int, int (*)(const char *, int),
106     old_glob_t *);
107 extern void old_globfree(old_glob_t *);

109 #define DOLLAR          '$'
110 #define DOT             '.'
111 #define EOS             '\0'
112 #define LBRACKET        '['
113 #define NOT             '!'
114 #define QUESTION        '?'
115 #define QUOTE           '\\'
116 #define RANGE           '-'
117 #define RBRACKET        ']'
118 #define SEP             '/'
119 #define STAR            '*'
120 #define TILDE           '~'
121 #define UNDERSCORE      '_'
122 #define LBRACE          '{'
123 #define RBRACE          '}'
124 #define SLASH           '/'
125 #define COMMA           ','
126 #define COLON           ':'

128 #define M_QUOTE         0x800000
129 #define M_PROTECT       0x400000

131 typedef struct wcat {
132         wchar_t w_wc;
133         uint_t w_at;
134 } wcat_t;

136 #define M_ALL           '*'     /* Plus M_QUOTE */
137 #define M_END           ']'     /* Plus M_QUOTE */
138 #define M_NOT           '!'     /* Plus M_QUOTE */
139 #define M_ONE           '?'     /* Plus M_QUOTE */
140 #define M_RNG           '-'     /* Plus M_QUOTE */
141 #define M_SET           '['     /* Plus M_QUOTE */
142 #define M_CLASS         ':'     /* Plus M_QUOTE */
143 #define ismeta(c)       (((c).w_at&M_QUOTE) != 0)

145 #define INITIAL                 8       /* initial pathv allocation */
 66 #define NULLCPP ((char **)0)    /* Null char ** */
 67 #define NAME_MAX        1024    /* something large */
```



```
147 #define GLOB_LIMIT_MALLOC       65536
148 #define GLOB_LIMIT_STAT         2048
149 #define GLOB_LIMIT_READDIR      16384
 69 static int      globit(size_t, const char *, glob_t *, int,
 70         int (*)(const char *, int), char **);
 71 static int      pstrcmp(const void *, const void *);
 72 static int      append(glob_t *, const char *);

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

154 struct glob_lim {
155         size_t  glim_malloc;
156         size_t  glim_stat;
157         size_t  glim_readdir;
158 };

160 struct glob_path_stat {
161         char            *gps_path;
162         struct stat     *gps_stat;
163 };

165 static int       compare(const void *, const void *);
166 static int       compare_gps(const void *, const void *);
167 static int       g_Ctoc(const wcat_t *, char *, uint_t);
168 static int       g_lstat(wcat_t *, struct stat *, glob_t *);
169 static DIR      *g_opendir(wcat_t *, glob_t *);
170 static wcat_t   *g_strchr(const wcat_t *, wchar_t);
171 static int       g_stat(wcat_t *, struct stat *, glob_t *);
172 static int       glob0(const wcat_t *, glob_t *, struct glob_lim *,
173                        int (*)(const char *, int));
174 static int       glob1(wcat_t *, wcat_t *, glob_t *, struct glob_lim *,
175                        int (*)(const char *, int));
176 static int       glob2(wcat_t *, wcat_t *, wcat_t *, wcat_t *, wcat_t *,
177                        wcat_t *, glob_t *, struct glob_lim *,
178                        int (*)(const char *, int));
179 static int       glob3(wcat_t *, wcat_t *, wcat_t *, wcat_t *, wcat_t *,
180                        wcat_t *, wcat_t *, glob_t *, struct glob_lim *,
181                        int (*)(const char *, int));
182 static int       globextend(const wcat_t *, glob_t *, struct glob_lim *,
183                           struct stat *);
184 static
185 const wcat_t    *globtilde(const wcat_t *, wcat_t *, size_t, glob_t *);
186 static int       globexp1(const wcat_t *, glob_t *, struct glob_lim *,
187                           int (*)(const char *, int));
188 static int       globexp2(const wcat_t *, const wcat_t *, glob_t *,
189                           struct glob_lim *, int (*)(const char *, int));
190 static int       match(wcat_t *, wcat_t *, wcat_t *, int);

192 /*
193  * Extended glob() function, selected by #pragma redefine_extname
194  * in glob.h with the external name _glob_ext() .
 75  * Free all space consumed by glob.
195  */
196 int
197 _glob_ext(const char *pattern, int flags, int (*errfunc)(const char *, int),
198     glob_t *pglob)
 77 void
 78 globfree(glob_t *gp)
199 {
200         const char *patnext;
201         int n;
202         size_t patlen;
203         wchar_t c;
204         wcat_t *bufnext, *bufend, patbuf[MAXPATHLEN];
```

```
205         struct glob_lim limit = { 0, 0, 0 };
 80         size_t i;

207         if ((patlen = strnlen(pattern, PATH_MAX)) == PATH_MAX)
208                 return (GLOB_NOMATCH);
 82         if (gp->gl_pathv == 0)
 83                 return;

210         patnext = pattern;
211         if (!(flags & GLOB_APPEND)) {
212                 pglob->gl_pathc = 0;
213                 pglob->gl_pathn = 0;
214                 pglob->gl_pathv = NULL;
215                 if ((flags & GLOB_KEEPSTAT) != 0)
216                         pglob->gl_statv = NULL;
217                 if (!(flags & GLOB_DOOFFS))
218                         pglob->gl_offs = 0;
219         }
220         pglob->gl_flags = flags & ~GLOB_MAGCHAR;
221         pglob->gl_matchc = 0;
 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);

223         if (pglob->gl_offs >= INT_MAX || pglob->gl_pathc >= INT_MAX ||
224             pglob->gl_pathc >= INT_MAX - pglob->gl_offs - 1)
225                 return (GLOB_NOSPACE);

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



```
265                         } else if (n == 0) {
266                                 break;
267                         } else {
268                                 return (GLOB_NOMATCH);
269                         }
270                 }
271         }
272         bufnext->w_at = 0;
273         bufnext->w_wc = EOS;

275         if (flags & GLOB_BRACE)
276                 return (globexp1(patbuf, pglob, &limit, errfunc));
277         else
278                 return (glob0(patbuf, pglob, &limit, errfunc));
 89         gp->gl_pathc = 0;
 90         gp->gl_pathv = NULLCPP;
279 }

281 /*
282  * Expand recursively a glob {} pattern. When there is no more expansion
283  * invoke the standard globbing routine to glob the rest of the magic
284  * characters
 94  * Do filename expansion.
285  */
286 static int
287 globexp1(const wcat_t *pattern, glob_t *pglob, struct glob_lim *limitp,
288     int (*errfunc)(const char *, int))
 96 int
 97 glob(const char *pattern, int flags,
 98         int (*errfn)(const char *, int), glob_t *gp)
289 {
290         const wcat_t *ptr = pattern;
100         int rv;
101         size_t i;
102         size_t ipathc;
103         char    *path;

292         /* Protect a single {}, for find(1), like csh */
293         if (pattern[0].w_wc == LBRACE && pattern[1].w_wc == RBRACE &&
294             pattern[2].w_wc == EOS)
295                 return (glob0(pattern, pglob, limitp, errfunc));
105         if ((flags & GLOB_DOOFFS) == 0)
106                 gp->gl_offs = 0;

297         if ((ptr = (const wcat_t *) g_strchr(ptr, LBRACE)) != NULL)
298                 return (globexp2(ptr, pattern, pglob, limitp, errfunc));
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);

300         return (glob0(pattern, pglob, limitp, errfunc));
301 }
113                 if (gp->gl_pathv == NULLCPP)
114                         return (GLOB_NOSPACE);
115                 gp->gl_pathp = gp->gl_pathv + gp->gl_offs;


304 /*
305  * Recursive brace globbing helper. Tries to expand a single brace.
306  * If it succeeds then it invokes globexp1 with the new pattern.
307  * If it fails then it tries to glob the rest of the pattern and returns.
308  */
309 static int
310 globexp2(const wcat_t *ptr, const wcat_t *pattern, glob_t *pglob,
311     struct glob_lim *limitp, int (*errfunc)(const char *, int))
```

```
312 {
313         int     i, rv;
314         wcat_t   *lm, *ls;
315         const wcat_t *pe, *pm, *pl;
316         wcat_t    patbuf[MAXPATHLEN];
317
318         /* copy part up to the brace */
319         for (lm = patbuf, pm = pattern; pm != ptr; *lm++ = *pm++)
320                 ;
321         lm->w_at = 0;
322         lm->w_wc = EOS;
323         ls = lm;
324
325         /* Find the balanced brace */
326         for (i = 0, pe = ++ptr; pe->w_wc != EOS; pe++)
327                 if (pe->w_wc == LBRACKET) {
328                         /* Ignore everything between [] */
329                         for (pm = pe++; pe->w_wc != RBRACKET &&
330                             pe->w_wc != EOS; pe++)
331                                 ;
332                         if (pe->w_wc == EOS) {
333                                 /*
334                                  * We could not find a matching RBRACKET.
335                                  * Ignore and just look for RBRACE
336                                  */
337                                 pe = pm;
117                 for (i = 0; i < gp->gl_offs; ++i)
118                         gp->gl_pathv[i] = NULL;
338                         }
339                 } else if (pe->w_wc == LBRACE) {
340                         i++;
341                 } else if (pe->w_wc == RBRACE) {
342                         if (i == 0)
343                                 break;
344                         i--;
345                 }
346
347         /* Non matching braces; just glob the pattern */
348         if (i != 0 || pe->w_wc == EOS)
349                 return (glob0(patbuf, pglob, limitp, errfunc));
121         if ((path = malloc(strlen(pattern)+1)) == NULL)
122                 return (GLOB_NOSPACE);
123
351         for (i = 0, pl = pm = ptr; pm <= pe; pm++) {
352                 switch (pm->w_wc) {
353                 case LBRACKET:
354                         /* Ignore everything between [] */
355                         for (pl = pm++; pm->w_wc != RBRACKET && pm->w_wc != EOS;
356                             pm++)
357                                 ;
358                         if (pm->w_wc == EOS) {
359                                 /*
360                                  * We could not find a matching RBRACKET.
361                                  * Ignore and just look for RBRACE
362                                  */
363                                 pm = pl;
364                         }
365                         break;
124         ipathc = gp->gl_pathc;
125         rv = globit(0, pattern, gp, flags, errfn, &path);
126
367                 case LBRACE:
368                         i++;
369                         break;
370
371                 case RBRACE:
```



```
372                         if (i) {
373                                 i--;
374                                 break;
375                         }
376                         /* FALLTHROUGH */
377                 case COMMA:
378                         if (i && pm->w_wc == COMMA)
379                                 break;
380                         else {
381                                 /* Append the current string */
382                                 for (lm = ls; (pl < pm); *lm++ = *pl++)
383                                         ;
384
127         if (rv == GLOB_ABORTED) {
385                                 /*
386                                  * Append the rest of the pattern after the
387                                  * 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.
388                                  */
389                                 for (pl = pe + 1;
390                                     (*lm++ = *pl++).w_wc != EOS; /* */)
391                                         ;
392
393                                 /* Expand the current pattern */
394                                 rv = globexp1(patbuf, pglob, limitp, errfunc);
395                                 if (rv && rv != GLOB_NOMATCH)
396                                         return (rv);
397
398                                 /* move after the comma, to the next string */
399                                 pl = pm + 1;
132                 free(path);
133                 return (GLOB_ABORTED);
400                         }
401                         break;
402
403                 default:
404                         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);
405                 }
406         }
407         return (0);
408 }
409
410
411
412 /*
413  * expand tilde from the passwd file.
414  */
415 static const wcat_t *
416 globtilde(const wcat_t *pattern, wcat_t *patbuf, size_t patbuf_len,
417     glob_t *pglob)
418 {
419         struct passwd *pwd;
420         char *h;
421         const wcat_t *p;
422         wcat_t *b, *eb, *q;
423         int n;
424         size_t lenh;
425         wchar_t c;
426
427         if (pattern->w_wc != TILDE || !(pglob->gl_flags & GLOB_TILDE))
428                 return (pattern);
```

```
430         /* Copy up to the end of the string or / */
431         eb = &patbuf[patbuf_len - 1];
432         for (p = pattern + 1, q = patbuf;
433             q < eb && p->w_wc != EOS && p->w_wc != SLASH; *q++ = *p++)
434                 ;
436         q->w_at = 0;
437         q->w_wc = EOS;
439         /* What to do if patbuf is full? */
441         if (patbuf[0].w_wc == EOS) {
442                 /*
443                  * handle a plain ~ or ~/ by expanding $HOME
444                  * first and then trying the password file
445                  */
446                 if (issetugid() != 0)
447                         return (pattern);
448                 if ((h = getenv("HOME")) == NULL) {
449                         if ((pwd = getpwuid(getuid())) == NULL)
450                                 return (pattern);
141         if (i == 0) {
142                 if (flags & GLOB_NOCHECK)
143                         (void) append(gp, pattern);
451                         else
452                                 h = pwd->pw_dir;
145                         rv = GLOB_NOMATCH;
453                 }
454         } else {
455                 /*
456                  * Expand a ~user
457                  */
458                 if ((pwd = getpwnam((char *)patbuf)) == NULL)
459                         return (pattern);
460                 else
461                         h = pwd->pw_dir;
462         }
147         gp->gl_pathp[gp->gl_pathc] = NULL;
148         free(path);
464         /* Copy the home directory */
465         lenh = strlen(h) + 1;
466         for (b = patbuf; b < eb && *h != EOS; b++) {
467                 if ((n = mbtowc(&c, h, lenh)) > 0) {
468                         h += n;
469                         lenh -= n;
470                         b->w_at = 0;
471                         b->w_wc = c;
472                 } else if (n < 0) {
473                         return (pattern);
474                 } else {
475                         break;
476                 }
477         }
479         /* Append the rest of the pattern */
480         while (b < eb && (*b++ = *p++).w_wc != EOS)
481                 ;
482         b->w_at = 0;
483         b->w_wc = EOS;
485         return (patbuf);
150         return (rv);
486 }
```



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

```
550         int a;
551         wcat_t *bufnext, patbuf[MAXPATHLEN];
161         size_t n;
162         size_t m;
163         ssize_t end = 0;        /* end of expanded directory */
164         char *pat = (char *)sp; /* pattern component */
165         char *dp = (*path) + dend;
166         int expand = 0;         /* path has pattern */
167         char *cp;
168         struct stat64 sb;
169         DIR *dirp;
170         struct dirent64 *d;
171         int err;

553         qpatnext = globtilde(pattern, patbuf, MAXPATHLEN, pglob);
554         oldpathc = pglob->gl_pathc;
555         bufnext = patbuf;

557         /*
558          * We don't need to check for buffer overflow any more.
559          * The pattern has already been copied to an internal buffer.
560          */
561         while ((a = qpatnext->w_at), (c = (qpatnext++)->w_wc) != EOS) {
562                 switch (c) {
563                 case LBRACKET:
564                         if (a != 0) {
565                                 bufnext->w_at = a;
566                                 (bufnext++)->w_wc = c;
567                                 break;
173         for (;;)
174                 switch (*dp++ = *(unsigned char *)sp++) {
175                 case '\0':      /* end of source path */
176                         if (expand)
177                                 goto Expand;
178                         else {
179                                 if (!(flags & GLOB_NOCHECK) ||
180                                     flags & (GLOB__CHECK|GLOB_MARK))
181                                         if (stat64(*path, &sb) < 0) {
182                                                 return (0);
568                         }
569                         a = qpatnext->w_at;
570                         c = qpatnext->w_wc;
571                         if (a == 0 && c == NOT)
572                                 ++qpatnext;
573                         if (qpatnext->w_wc == EOS ||
574                             g_strchr(qpatnext+1, RBRACKET) == NULL) {
575                                 bufnext->w_at = 0;
576                                 (bufnext++)->w_wc = LBRACKET;
577                                 if (a == 0 && c == NOT)
578                                         --qpatnext;
579                                 break;
184                                 if (flags & GLOB_MARK && S_ISDIR(sb.st_mode)) {
185                                         *dp = '\0';
186                                         *--dp = '/';
580                         }
581                         bufnext->w_at = M_QUOTE;
582                         (bufnext++)->w_wc = M_SET;
583                         if (a == 0 && c == NOT) {
584                                 bufnext->w_at = M_QUOTE;
585                                 (bufnext++)->w_wc = M_NOT;
586                         }
587                         a = qpatnext->w_at;
588                         c = (qpatnext++)->w_wc;
589                         do {
590                                 if (a == 0 && c == LBRACKET &&
591                                     qpatnext->w_wc == COLON) {
```



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

```
658                         bufnext->w_at = a;
659                         (bufnext++)->w_wc = c;
660                         break;
661                 }
662         }
663         bufnext->w_at = 0;
664         bufnext->w_wc = EOS;

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

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

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

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



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

720 static int
721 compare_gps(const void *_p, const void *_q)
722 {
723         const struct glob_path_stat *p = (const struct glob_path_stat *)_p;
724         const struct glob_path_stat *q = (const struct glob_path_stat *)_q;
202                 case '/':
203                         if (expand)
204                                 goto Expand;
205                         end = dp - *path;
206                         pat = (char *)sp;
207                         break;

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

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

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

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

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

766                         if ((pglob->gl_flags & GLOB_LIMIT) &&
767                             limitp->glim_stat++ >= GLOB_LIMIT_STAT) {
768                                 errno = 0;
769                                 pathend->w_at = 0;
770                                 (pathend++)->w_wc = SEP;
771                                 pathend->w_at = 0;
772                                 pathend->w_wc = EOS;
773                                 return (GLOB_NOSPACE);
209                 Expand:
```

```
210                     /* determine directory and open it */
211                     (*path)[end] = '\0';
212                     dirp = opendir(**path == '\0' ? "." : *path);
213                     if (dirp == NULL) {
214                             if (errfn != 0 && errfn(*path, errno) != 0 ||
215                                 flags&GLOB_ERR) {
216                                     return (GLOB_ABORTED);
774                     }
775                     if (g_lstat(pathbuf, &sb, pglob))
776                             return (0);

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

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

807             if (!anymeta) {         /* No expansion, do next segment. */
808                     pathend = q;
809                     pattern = p;
810                     while (pattern->w_wc == SEP) {
811                             if (pathend+1 > pathend_last)
232                     /* expand path to max. expansion */
233                     n = dp - *path;
234                     *path = realloc(*path,
235                         strlen(*path) + NAME_MAX + strlen(sp) + 1);
236                     if (*path == NULL) {
237                             (void) closedir(dirp);
238                             free(pat);
812                                     return (GLOB_NOSPACE);
813                             *pathend++ = *pattern++;
814                     }
815             } else  {
816                     /* Need expansion, recurse. */
817                     return (glob3(pathbuf, pathbuf_last, pathend,
```



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

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

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

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

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

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

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

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

```
882                         pathend->w_wc = EOS;
883                         err = GLOB_NOSPACE;
884                         break;
885                 }
886
887                 /* Initial DOT must be matched literally. */
888                 if (dp->d_name[0] == DOT && pattern->w_wc != DOT)
245                         while ((d = readdir64(dirp)) != NULL) {
246                                 cp = d->d_name;
247                                 if ((flags&GLOB_NOESCAPE)
248                                     ? fnmatch(pat, cp, FNM_PERIOD|FNM_NOESCAPE)
249                                     : fnmatch(pat, cp, FNM_PERIOD))
889                         continue;
890                 dc = pathend;
891                 sc = dp->d_name;
892                 lensc = strlen(sc) + 1;
893                 while (dc < pathend_last) {
894                         if ((n = mbtowc(&w, sc, lensc)) <= 0) {
895                                 sc += 1;
896                                 lensc -= 1;
897                                 dc->w_at = 0;
898                                 dc->w_wc = EOS;
899                         } else {
900                                 sc += n;
901                                 lensc -= n;
902                                 dc->w_at = 0;
903                                 dc->w_wc = w;
904                         }
905                         dc++;
906                         if (n <= 0)
907                                 break;
908                 }
909                 if (dc >= pathend_last) {
910                         dc->w_at = 0;
911                         dc->w_wc = EOS;
912                         err = GLOB_NOSPACE;
913                         break;
914                 }
915                 if (n < 0) {
916                         err = GLOB_NOMATCH;
917                         break;
918                 }

920                 if (!match(pathend, pattern, restpattern, GLOB_LIMIT_RECUR)) {
921                         pathend->w_at = 0;
922                         pathend->w_wc = EOS;
923                         continue;
924                 }
925                 err = glob2(pathbuf, pathbuf_last, --dc, pathend_last,
926                     restpattern, restpattern_last, pglob, limitp,
927                     errfunc);
928                 if (err)
252                         n = strlen(cp);
253                         (void) memcpy((*path) + end, cp, n);
254                         m = dp - *path;
255                         err = globit(end+n, sp, gp, flags, errfn, path);
256                         dp = (*path) + m;   /* globit can move path */
257                         if (err != 0)
929                         break;
930         }

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



```
936         return (err);
937 }


940 /*
941  * Extend the gl_pathv member of a glob_t structure to accommodate a new item,
942  * add the new item, and update gl_pathc.  Avoids excessive reallocation
943  * by doubling the number of elements each time.  Uses gl_pathn to contain
944  * the number.
945  *
946  * Return 0 if new item added, error code if memory couldn't be allocated.
947  *
948  * Invariant of the glob_t structure:
949  *      Either gl_pathc is zero and gl_pathv is NULL; or gl_pathc > 0 and
950  *      gl_pathv points to (gl_offs + gl_pathc + 1) items.
951  */
952 static int
953 globextend(const wcat_t *path, glob_t *pglob, struct glob_lim *limitp,
954     struct stat *sb)
955 {
956         char **pathv;
957         ssize_t i;
958         size_t allocn, newn, len;
959         char *copy = NULL;
960         const wcat_t *p;
961         struct stat **statv;
962         char junk[MB_LEN_MAX];
963         int n;

965         allocn = pglob->gl_pathn;
966         newn = 2 + pglob->gl_pathc + pglob->gl_offs;

968         if (newn <= allocn) {
969                 pathv = pglob->gl_pathv;
970                 if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0)
971                         statv = pglob->gl_statv;
972         } else {
973                 if (allocn == 0)
974                         allocn = pglob->gl_offs + INITIAL;
975                 allocn *= 2;
976                 if (pglob->gl_offs >= INT_MAX ||
977                     pglob->gl_pathc >= INT_MAX ||
978                     allocn >= INT_MAX ||
979                     SIZE_MAX / sizeof (*pathv) <= allocn ||
980                     SIZE_MAX / sizeof (*statv) <= allocn) {
981                 nospace:
982                         for (i = pglob->gl_offs; i < (ssize_t)(newn - 2);
983                             i++) {
984                                 if (pglob->gl_pathv && pglob->gl_pathv[i])
985                                         free(pglob->gl_pathv[i]);
986                                 if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
987                                     pglob->gl_statv && pglob->gl_statv[i])
988                                         free(pglob->gl_statv[i]);
989                         }
990                         if (pglob->gl_pathv) {
991                                 free(pglob->gl_pathv);
992                                 pglob->gl_pathv = NULL;
993                         }
994                         if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0 &&
995                             pglob->gl_statv) {
996                                 free(pglob->gl_statv);
997                                 pglob->gl_statv = NULL;
998                         }
999                         return (GLOB_NOSPACE);
1000                }
1001                limitp->glim_malloc += allocn * sizeof (*pathv);
```

```
1002                pathv = realloc(pglob->gl_pathv, allocn * sizeof (*pathv));
1003                if (pathv == NULL)
1004                        goto nospace;
1005                if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0) {
1006                        limitp->glim_malloc += allocn * sizeof (*statv);
1007                        statv = realloc(pglob->gl_statv,
1008                            allocn * sizeof (*statv));
1009                        if (statv == NULL)
1010                                goto nospace;
1011                }
1012        }
1013        pglob->gl_pathn = allocn;

1015        if (pglob->gl_pathv == NULL && pglob->gl_offs > 0) {
1016                /* first time around -- clear initial gl_offs items */
1017                pathv += pglob->gl_offs;
1018                for (i = pglob->gl_offs; --i >= 0; )
1019                        *--pathv = NULL;
1020        }
1021        pglob->gl_pathv = pathv;

1023        if ((pglob->gl_flags & GLOB_KEEPSTAT) != 0) {
1024                if (pglob->gl_statv == NULL && pglob->gl_offs > 0) {
1025                        /* first time around -- clear initial gl_offs items */
1026                        statv += pglob->gl_offs;
1027                        for (i = pglob->gl_offs; --i >= 0; )
1028                                *--statv = NULL;
1029                }
1030                pglob->gl_statv = statv;
1031                if (sb == NULL)
1032                        statv[pglob->gl_offs + pglob->gl_pathc] = NULL;
1033                else {
1034                        limitp->glim_malloc += sizeof (**statv);
1035                        if ((statv[pglob->gl_offs + pglob->gl_pathc] =
1036                            malloc(sizeof (**statv))) == NULL)
1037                                goto copy_error;
1038                        (void) memcpy(statv[pglob->gl_offs + pglob->gl_pathc],
1039                            sb, sizeof (*sb));
1040                }
1041                statv[pglob->gl_offs + pglob->gl_pathc + 1] = NULL;
1042        }

1044        len = MB_LEN_MAX;
1045        p = path;
1046        while ((n = wctomb(junk, p->w_wc)) > 0) {
1047                len += n;
1048                if ((p++)->w_wc == EOS)
1049                        break;
1050        }
1051        if (n < 0)
1052                return (GLOB_NOMATCH);

1054        limitp->glim_malloc += len;
1055        if ((copy = malloc(len)) != NULL) {
1056                if (g_Ctoc(path, copy, len)) {
1057                        free(copy);
1058                        return (GLOB_NOSPACE);
1059                }
1060                pathv[pglob->gl_offs + pglob->gl_pathc++] = copy;
1061        }
1062        pathv[pglob->gl_offs + pglob->gl_pathc] = NULL;

1064        if ((pglob->gl_flags & GLOB_LIMIT) &&
1065            limitp->glim_malloc >= GLOB_LIMIT_MALLOC) {
1066                errno = 0;
1067                return (GLOB_NOSPACE);
```



```
1068        }
1069        copy_error:
1070        return (copy == NULL ? GLOB_NOSPACE : 0);
 265                /* NOTREACHED */
1071 }


1074 /*
1075  * pattern matching function for filenames.  Each occurrence of the *
1076  * pattern causes a recursion level.
 269  * Comparison routine for two name arguments, called by qsort.
1077  */
1078 static int
1079 match(wcat_t *name, wcat_t *pat, wcat_t *patend, int recur)
 271 int
 272 pstrcmp(const void *npp1, const void *npp2)
1080 {
1081        int ok, negate_range;
1082        wcat_t c, k;

1084        if (recur-- == 0)
1085                return (1);

1087        while (pat < patend) {
1088                c = *pat++;
1089                switch (c.w_wc) {
1090                case M_ALL:
1091                        if (c.w_at != M_QUOTE) {
1092                                k = *name++;
1093                                if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1094                                        return (0);
1095                                break;
1096                        }
1097                        while (pat < patend && pat->w_at == M_QUOTE &&
1098                            pat->w_wc == M_ALL)
1099                                pat++;  /* eat consecutive '*' */
1100                        if (pat == patend)
1101                                return (1);
1102                        do {
1103                                if (match(name, pat, patend, recur))
1104                                        return (1);
1105                        } while ((name++)->w_wc != EOS);
1106                        return (0);
1107                case M_ONE:
1108                        if (c.w_at != M_QUOTE) {
1109                                k = *name++;
1110                                if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1111                                        return (0);
1112                                break;
1113                        }
1114                        if ((name++)->w_wc == EOS)
1115                                return (0);
1116                        break;
1117                case M_SET:
1118                        if (c.w_at != M_QUOTE) {
1119                                k = *name++;
1120                                if (k.w_at != c.w_at || k.w_wc != c.w_wc)
1121                                        return (0);
1122                                break;
1123                        }
1124                        ok = 0;
1125                        if ((k = *name++).w_wc == EOS)
1126                                return (0);
1127                        if ((negate_range = (pat->w_at == M_QUOTE &&
1128                            pat->w_wc == M_NOT)) != 0)
1129                                ++pat;
```

```
                        while (((c = *pat++).w_at != M_QUOTE) ||
                            c.w_wc != M_END) {
                                if (c.w_at == M_QUOTE && c.w_wc == M_CLASS) {
                                        wcat_t cc;

                                        cc.w_at = pat->w_at;
                                        cc.w_wc = pat->w_wc;
                                        if (iswctype(k.w_wc, cc.w_wc))
                                                ok = 1;
                                        ++pat;
                                }
                                if (pat->w_at == M_QUOTE &&
                                    pat->w_wc == M_RNG) {
                                        if (c.w_wc <= k.w_wc &&
                                            k.w_wc <= pat[1].w_wc)
                                                ok = 1;
                                        pat += 2;
                                } else if (c.w_wc == k.w_wc)
                                        ok = 1;
                        }
                        if (ok == negate_range)
                                return (0);
                        break;
                default:
                        k = *name++;
                        if (k.w_at != c.w_at || k.w_wc != c.w_wc)
                                return (0);
                        break;
                }
        }
        return (name->w_wc == EOS);
        return (strcoll(*(char **)npp1, *(char **)npp2));
}

/*
 * Extended globfree() function, selected by #pragma redefine_extname
 * in glob.h with the external name _globfree_ext() .
 * Add a new matched filename to the glob_t structure, increasing the
 * size of that array, as required.
 */
void
_globfree_ext(glob_t *pglob)
int
append(glob_t *gp, const char *str)
{
        int i;
        char **pp;
        char *cp;

        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]);
                }
                free(pglob->gl_statv);
                pglob->gl_statv = NULL;
        }
```



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

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

        if (str->w_wc == EOS)
                (void) strlcpy(buf, ".", sizeof (buf));
        else {
                if (g_Ctoc(str, buf, sizeof (buf)))
                        return (NULL);
        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;
        }

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

        return (opendir(buf));
}

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

        if (g_Ctoc(fn, buf, sizeof (buf)))
                return (-1);
        if (pglob->gl_flags & GLOB_ALTDIRFUNC)
                return ((*pglob->gl_lstat)(buf, sb));
        return (lstat(buf, sb));
}

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

        if (g_Ctoc(fn, buf, sizeof (buf)))
                return (-1);
        if (pglob->gl_flags & GLOB_ALTDIRFUNC)
                return ((*pglob->gl_stat)(buf, sb));
        return (stat(buf, sb));
}

static wcat_t *
g_strchr(const wcat_t *str, wchar_t ch)
{
        do {
                if (str->w_at == 0 && str->w_wc == ch)
                        return ((wcat_t *)str);
        } while ((str++)->w_wc != EOS);
        return (NULL);
}

static int
g_Ctoc(const wcat_t *str, char *buf, uint_t len)
```

```
1246 {
1247         int n;
1248         wchar_t w;

1250         while (len >= MB_LEN_MAX) {
1251                 w = (str++)->w_wc;
1252                 if ((n = wctomb(buf, w)) > 0) {
1253                         len -= n;
1254                         buf += n;
1255                 }
1256                 if (n < 0)
1257                         break;
1258                 if (w == EOS)
1259                         return (0);
1260         }
1261         return (1);
1262 }

1264 /* glob() function with legacy glob structure */
1265 int
1266 old_glob(const char *pattern, int flags, int (*errfunc)(const char *, int),
1267     old_glob_t *pglob)
1268 {

1270         glob_t gl;
1271         int rv;

1273         flags &= GLOB_POSIX;

1275         (void) memset(&gl, 0, sizeof (gl));

1277         /*
1278          * Copy all the members, old to new.  There's
1279          * really no point in micro-optimizing the copying.
1280          * Other members are set to zero.
1281          */
1282         gl.gl_pathc = pglob->gl_pathc;
1283         gl.gl_pathv = pglob->gl_pathv;
1284         gl.gl_offs = pglob->gl_offs;
1285         gl.gl_pathp = pglob->gl_pathp;
1286         gl.gl_pathn = pglob->gl_pathn;

1288         rv = _glob_ext(pattern, flags, errfunc, &gl);

1290         /*
1291          * Copy all the members, new to old.  There's
1292          * really no point in micro-optimizing the copying.
1293          */
1294         pglob->gl_pathc = gl.gl_pathc;
1295         pglob->gl_pathv = gl.gl_pathv;
1296         pglob->gl_offs = gl.gl_offs;
1297         pglob->gl_pathp = gl.gl_pathp;
1298         pglob->gl_pathn = gl.gl_pathn;

1300         return (rv);
1301 }

1303 /* globfree() function with legacy glob structure */
1304 void
1305 old_globfree(old_glob_t *pglob)
1306 {
1307         glob_t gl;

1309         (void) memset(&gl, 0, sizeof (gl));

1311         /*
```



```
1312          * Copy all the members, old to new.  There's
1313          * really no point in micro-optimizing the copying.
1314          * Other members are set to zero.
1315          */
1316         gl.gl_pathc = pglob->gl_pathc;
1317         gl.gl_pathv = pglob->gl_pathv;
1318         gl.gl_offs = pglob->gl_offs;
1319         gl.gl_pathp = pglob->gl_pathp;
1320         gl.gl_pathn = pglob->gl_pathn;

1322         _globfree_ext(&gl);

1324         /*
1325          * Copy all the members, new to old.  There's
1326          * really no point in micro-optimizing the copying.
1327          */
1328         pglob->gl_pathc = gl.gl_pathc;
1329         pglob->gl_pathv = gl.gl_pathv;
1330         pglob->gl_offs = gl.gl_offs;
1331         pglob->gl_pathp = gl.gl_pathp;
1332         pglob->gl_pathn = gl.gl_pathn;

1334 }

1336 /* End */
```

```
**********************************************************
   17719 Thu Oct 10 15:32:58 2013
new/usr/src/man/man3c/glob.3c
1097 glob(3c) needs to support non-POSIX options
3341 The sftp command should use the native glob()
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright (c) 1992, X/Open Company Limited. All Rights Reserved.
   3 .\" Portions Copyright (c) 2003, Sun Microsystems, Inc.  All Rights Reserved.
   4 .\" Portions Copyright (c) 2013, Gary Mills
   2 .\" Copyright (c) 1992, X/Open Company Limited. All Rights Reserved.  Portions C
   5 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
   6 .\" http://www.opengroup.org/bookstore/.
   7 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
   8 .\"
   9 .\" $OpenBSD: glob.3,v 1.30 2012/01/20 07:09:42 tedu Exp $
  10 .\"
  11 .\" Copyright (c) 1989, 1991, 1993, 1994
  12 .\" The Regents of the University of California.  All rights reserved.
  13 .\"
  14 .\" This code is derived from software contributed to Berkeley by
  15 .\" Guido van Rossum.
  16 .\" Redistribution and use in source and binary forms, with or without
  17 .\" modification, are permitted provided that the following conditions
  18 .\" are met:
  19 .\" 1. Redistributions of source code must retain the above copyright
  20 .\"    notice, this list of conditions and the following disclaimer.
  21 .\" 2. Redistributions in binary form must reproduce the above copyright
  22 .\"    notice, this list of conditions and the following disclaimer in the
  23 .\"    documentation and/or other materials provided with the distribution.
  24 .\" 3. Neither the name of the University nor the names of its contributors
  25 .\"    may be used to endorse or promote products derived from this software
  26 .\"    without specific prior written permission.
  27 .\"
  28 .\" THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
  29 .\" ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  30 .\" IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
  31 .\" ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
  32 .\" FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  33 .\" DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  34 .\" OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  35 .\" HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
  36 .\" LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
  37 .\" OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
  38 .\" SUCH DAMAGE.
  39 .\"
  40 .\"  This notice shall appear on any product containing this material.
  41 .\" The contents of this file are subject to the terms of the Common Development
  42 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
  43 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
  44 .TH GLOB 3C "Nov 1, 2003"
  45 .SH NAME
  46 glob, globfree \- generate path names matching a pattern
  47 .SH SYNOPSIS
  48 .LP
  49 .nf
  50 #include <glob.h>

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

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



```
  60 .fi

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

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

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

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

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

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

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

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

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



```
187 .RE

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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

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



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

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

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

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

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

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

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

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



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

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

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

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

.sp
.LP
Using the same example:

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

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

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

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

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

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

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