```
int accept(int sock, struct sockaddr *addr, int addrlen);
int bind(int sock, struct sockaddr *addr, int addrlen);
int close(int fd);
int closedir(DIR *dir);
int connect(int sock, struct sockaddr *addr, int addrlen);
int dup2(int oldfd, int newfd);
int execlp(const char *file, char *argv0, ..., (char *)0);
int execvp(const char *file, char *argv[]);
int fclose(FILE *stream);
int FD_ISSET(int fd, fd_set *fds);
void FD_SET(int fd, fd_set *fds);
void FD_CLR(int fd, fd_set *fds);
void FD_ZERO(fd_set *fds);
int fgetc(FILE *stream);
char *fgets(char *s, int n, FILE *stream);
int fileno(FILE *stream);
pid_t fork(void);
FILE *fopen(const char *file, const char *mode);
int fprintf(FILE *stream, const char *format, ...);
int getchar(void);
struct hostent *gethostbyname(const char *name);
unsigned long int htonl(unsigned long int hostlong);
unsigned short int htons(unsigned short int hostshort);
char *index(const char *s, int c);
int kill(int pid, int signo);
int listen(int sock, int n);
unsigned long int ntohl(unsigned long int netlong);
unsigned short int ntohs(unsigned short int netshort);
int open(const char *path, int oflag); /* O_RDONLY, O_WRONLY or O_RDWR */
DIR *opendir(const char *name);
int pclose(FILE *stream);
int pipe(int filedes[2]);
FILE *popen(char *cmdstr, char *mode);
int putchar(int c);
ssize_t read(int fd, void *buf, size_t count);
struct dirent *readdir(DIR *dir);
int select(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds,
           struct timeval *timeout);
int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
    /* actions include SIG_DFL and SIG_IGN */
int sigaddset(sigset_t *set, int signum);
int sigemptyset(sigset_t *set);
int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);
    /* how has the value SIG_BLOCK, SIG_UNBLOCK, or SIG_SETMASK */
unsigned int sleep(unsigned int seconds);
int socket(int family, int type, int protocol);
    /* family is PF_UNIX, PF_INET; type is SOCK_STREAM, SOCK_DGRAM */
int sprintf(char *s, const char *format, ...);
```

```
int stat(const char *file name, struct stat *buf);
char *strchr(const char *s, int c);
size_t strlen(const char *s);
char *strncat(char *dest, const char *src, size_t n);
int strncmp(const char *s1, const char *s2, size_t n);
char *strncpy(char *dest, const char *src, size_t n);
char *strrchr(const char *s, int c);
char *strstr(const char *haystack, const char *needle);
int wait(int *status);
int waitpid(int pid, int *stat, int options); /* options = 0 or WNOHANG*/
ssize_t write(int fd, const void *buf, size_t count);

WIFEXITED(status)       WIFSIGNALED(status)     WIFSTOPPED(status)
WEXITSTATUS(status)     WTERMSIG(status)        WSTOPSIG(status)

struct hostent {
   char *h_name;     /* official name of host */
   char **h_aliases; /* alias list */
   int h_addrtype;   /* host address type */
   int h_length;     /* length of address */
   char *h_addr;     /* address */
};

struct sigaction {
   void (*sa_handler)(int);   /* function or SIG_DFL or SIG_IGN */
   sigset_t sa_mask;
   int sa_flags;     /* SA_NOCLDSTOP, SA_RESTART, SA_NOMASK, etc. */
};

struct sockaddr_in {
   sa_family_t    sin_family; /* AF_INET */
   u_int16_t      sin_port;
   struct in_addr sin_addr;
   unsigned char  sin_zero[8]; /*Unused*/
};
```

Shell variables:

`$$` shell process ID
`$?` last program exit status
`$#` number of arguments
`$*` all arguments as string
`"$@"` all arguments as quoted list

Shell test comparison operators:

Shell	Description
-d filename	Exists as a directory
-f filename	Exists as a regular file
-r filename	Exists as a readable file
-w filename	Exists as a writable file
-x filename	Exists as an executable file
-z string	True if empty string
str1 = str2	True if str1 equals str2
str1 != str2	True if str1 not equal to str2
int1 -eq int2	True if int1 equals int2
-ne, -gt, -ge, -lt, -le	Comparisons for numbers
!=, >, >=, <, <=	Comparisons for strings
-a, -o	And, or