

C function prototypes and structs:

```

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)
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, ...)
struct hostent *gethostbyname(const char *name)
unsigned long int htonl(unsigned long int hostlong);
unsigned short int htons(unsigned short int hostshort);
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)
DIR *opendir(const char *name);
int pclose(FILE *stream)
int pipe(int filedes[2])
FILE *popen(char *cmdstr, char *mode)
struct dirent *readdir(DIR *dir);
ssize_t Readline(int filedes, void *buf, size_t maxlen);
ssize_t Readn(int filedes, void *buf, size_t nbytes);
int select(int maxfdp1, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout)
int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
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)
int sprintf(char *s, const char *format, ...)
int stat(const char *file_name, struct stat *buf);
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);
int wait(int *status)
int waitpid(int pid, int *stat, int options) /* options = 0 or WNOHANG*/
void Writen(int filedes, const void *buf, size_t nbytes);

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

struct sigaction {
    void (*sa_handler)(int);
    sigset_t sa_mask;
    int sa_flags; /*0*/
}

struct sockaddr_in {
    sa_family_t sin_family;
    struct in_addr sin_addr;
    unsigned char pad[8]; /*Unused*/
}

struct hostent {
    char *h_name; /* official name */
    char **h_aliases; /* alias list */

    int h_addrtype; /* host address type */
    int h_length; /* length of address */
    char *h_addr; /*address*/
}

```

Shell and Perl comparison operators

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

Perl functions:

push(ARRAY, LIST)
 pop(ARRAY) -- returns LIST
 sort BLOCK LIST -- returns LIST
 defined(SCALAR) -- returns true if SCALAR is defined, false otherwise
 split(/PATTERN/, SCALAR) - returns LIST
 open(FILEHANDLE, EXPR) -- return true if filename given by EXPR is opened

Meta characters that need to be escaped are \ | () [{ . \$ ^ ? +

Perl pattern matching:

\s = [\t\n\r\f]	space
\w = [a-zA-Z0-9_]	word
\d = [0-9]	digit
[]	one character of a set
[^]	any expect one of the set
+	one or more
*	zero or more
?	zero or one
.	any character

Perl Special variables

@_	arguments to a subroutine
\$_	the default scalar variable
\$.	the current input line number
\$0	program name
\$\$	process id
#!	last system call error
@ARGV	command line arguments