CSC209 April 2019: Aid Sheet

C function prototypes

```
int accept(int sock, struct sockaddr *addr, int *addrlen);
int bind(int sock, struct sockaddr *addr, int addrlen);
int close(int fd);
int connect(int sock, struct sockaddr *addr, int addrlen);
int dup2(int oldfd, int newfd);
int execl(const char *path, const char *arg0, ...  /*, (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 * restrict stream, const char * restrict format, ...);
size_t fread(void *ptr, size_t size, size_t nitems, FILE *stream);
      /* returns the number of items read */
void free(void *ptr);
int fseek(FILE *stream, long offset, int whence);
      /* whence has the value SEEK_SET, SEEK_CUR, or SEEK_END */
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
pid_t getpid(void);
pid_t getppid(void);
unsigned long int htonl(unsigned long int hostlong); /* 4 bytes */
unsigned short int htons(unsigned short int hostshort); /* 2 bytes */
int kill(int pid, int signo);
int listen(int sock, int n);
void *malloc(size_t size);
int open(const char *path, int oflag);
       /* oflag is O_WRONLY | O_CREAT for write and O_RDONLY for read */
int pipe(int filedes[2]);
ssize_t read(int d, void *buf, size_t nbytes);
      /* returns number of bytes read */
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);
       /* actions include SIG_DFL and SIG_IGN */
int sigaddset(sigset_t *set, int signum);
int sigemptyset(sigset_t *set);
int socket(int family, int type, int protocol);
       /* family = PF_INET, type = SOCK_STREAM, protocol = 0 */
int sprintf(char *s, const char *format, ...);
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);
long strtol(const char *restrict str, char **restrict endptr, int base);
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 d, const void *buf, size_t nbytes);
```

CSC209 April 2019: Aid Sheet

Excerpt from strcpy/strncpy man page:

The stpcpy() and strcpy() functions copy the string src to dst (including the terminating '\0' character).

The stpncpy() and strncpy() functions copy at most n characters from src into dst. If src is less than n characters long, the remainder of dst is filled with '\0' characters. Otherwise, dst is not terminated.

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

Useful structs

```
struct sigaction {
    void (*sa_handler)(int);
    sigset_t sa_mask;
    int sa_flags;
};
struct sockaddr_in {
    sa_family_t sin_family;
    unsigned short int sin_port;
    struct in_addr sin_addr;
    unsigned char pad[8]; /* Unused */
};
```

```
1)
2) sa.sa_handler
3) sa.flag = 0
4) sigemptyset(sa.sa_mask);
5) sigaction(SIGINT, &sa, NULL)
addr.sin_family = AF_INET;
addr.sin_port = htos(5432);
addr.sin_addr.s_addr = INADDR_ANY;
memset(&addr.sin_zero), 0, 8);
```

Shell 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, -lt, -le	For numbers
!=, >, >=, <, <=	For strings
-a, -o	And, or.

Useful Makefile variables:

$@	target
$^	list of prerequisites
$<	first prerequisite
$?	return code of last program executed

FD_ZERO

FD_SET

FD_ISSET

Useful shell commands:

cat, cd, chmod, cp, echo, expr, ls, mkdir, read, uniq, set
cut (-d delimeter -f field) (fields count from 1)
diff (returns 0 if the files are the same, and 1 if the files differ)
expr ARG1 + ARG2
grep (returns 0 if match is found, 1 if no match was found, and 2 if there was an error)
head (-n count, or default is 10)
seq n (prints the numbers from 1 to n inclusive) sort (sort lines of text files)
wc (-clw options return the number of characters, lines, and words respectively)

$0	Script name
$#	Number of positional parameters
$*	List of all positional parameters
$?	Exit value of previously executed command