# Handling input

## Parsing optional arguments

Use `:` with `getopt(..)` to denote options with an argument. Otherwise, options are read without arguments. The order of arguments is irrelevant.

```
while ((opt = getopt(argc, argv, "s:a")) != -1) {
      switch (opt) {
      case 's':
          // Handle argument s
          break;
      case 'x':
          // Handle argument for x
          break;
      default:
          // Handle other arguments
          break;
      }
  }
```

Use `optarg` to access current optional argument. Use `optind` to retrieve the current optional argument index.

## Parsing mandatory arguments

Assume that the program handles input of the form `-s 0 foo bar`. As neither foo nor bar are arguments, their existence must be validated without `getopt`. Use `optind` and `argc` to validate number of remaining parameters.

```
if(argc - optind != 2){
  usage();
}
```

## Printing usage

Use `stderr, stdout` as output stream. Use `[..]` to denote optional parameters. Use `|` to denote choice. Use `<...>` to denote mandatory parameters. Format strings using `%s, %d, %f` for strings, integers or floats respectively.

```
static void usage(void)
{
  fprintf(stderr, "SYNOPSIS: %s [-s | -a NUM] <string>\n", command);
  exit(EXIT_FAILURE);
}
```

## Server-Client

### Default imports

```c
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <resolv.h>
#include <netdb.h>
#include <errno.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

### Creating a socket

```c
int fd;
struct sockaddr_in sin;
sin.sin_family = AF_INET;
sin.sin_port = htons (port);
/* Automatically bind to local address (server) */
sin.sin_addr.s_addr = INADDR_ANY;
/* or specify one (client) */
//inet_pton(AF_INET, "127.0.0.1", &sin.sin_addr);
/* Create the socket */
fd = socket(AF_INET, SOCK_STREAM ,0);
/* Assign the address to the socket */
bind(fd, (struct sockaddr*)&sin, sizeof sin);
freeaddrinfo(ai);
```

### Creating a socket (protocol independent)

```c
struct addrinfo hints;
struct addrinfo *ai;
int fd,res;
memset (&hints, 0, sizeof(hints));
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
hints.ai_flags = AI_PASSIVE;
/* Bind to local adress with NULL */
res =getaddrinfo("127.0.0.1", port, &hints, &ai);
/* Create the socket */
fd = socket(ai->ai_family, ai->ai_socktype, ai->ai_protocol);
/* Assign the address to the socket */
res = bind(fd, ai->ai_addr, ai->ai_addrlen);
freeaddrinfo(ai);
```

### Reuse socket

If you get `Address already in use`, consider setting socket options.

```c
int optval = 1;
setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, &optval, sizeof(optval));
```

### Connecting to remote socket

```
struct addrinfo hints;
struct addrinfo *ai;
int fd, res;
memset (&hints, 0, sizeof(hints);
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;

res = getaddrinfo ("127.0.0.1", port, &hints, &ai);
/* Create the socket */
fd = socket (ai->ai_family, ai->ai_socktype, ai->ai_protocol);
/* Connect the socket to the address */
res = connect (fd, ai->ai_addr, ai->ai_addrlen);
freeaddrinfo(ai);
```

### Accepting a new connection

```
int cdf;
res = listen(fd, 5); // Listen on socket fd with backlog 5
cfd = accept(fd, (struct sockaddr *)&dest, &addr_size); //Open
```

### Sending a message

Send a message using the socket specified by `fd`. Alternatively use `send(..)` with flags `MSG_WAITALL, MSG_DONTWAIT`.

```
char request[32]; memset(&request, 0x0, 32);
strcpy(&request[0], "message");
res = write(fd, &request[0], 32);
```

### Receiving a message

Receive a message using `read`. Alternatively use `recv(...)`.

```
char buf[32]; memset(&buf, 0x0, 32);
res = read(cfd, &buf[0], 32);
```

For filling a bigger buffer with multiple smaller messages use this:

```
ssize_t num;
size_t total;
char buf[n];
for (total =0; total <n;) {
  num = read (cfd, &buf[total], n-total);
  total += num;
}
```

### Man pages

Use `man 7 ip` for documentation regarding `sockaddr_in`. Use `man 3 getaddrinfo` for documentation regarding `addrinfo`. Use `man -wK expression` or `man -k expression` to find man pages for expression. For example, `man -k addrinfo` yields

```
freeaddrinfo (3)      - network address and service translation
gai.conf (5)          - getaddrinfo(3) configuration file
getaddrinfo (3)       - network address and service translation
getaddrinfo_a (3)     - asynchronous network address and service translation
```

## Various snippets & Tricks

### Casting values

```
1 int i = strtol(string, (char**) NULL, 10); //Convert string to int
  char str[10]; sprintf(str, "%d", 10); // Convert integer to string
```