CSC 3600 Cheat Sheet

by Haskell via cheatography.com/57281/cs/15140/

Useful Functions

int socket(int domain, int type, int protocol);	Creates a socket and returns a file descriptor
struct hostent *gethostbyname(const char* name);	Checks if the host exists and then translates its address into something useable by other functions. Note: The data you probably want from this is contained in the h_addr field of the struct
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);	Attempts to connect
ssize_t send(int sockfd, const void *buf, size_t len, int flags);	Sends the specified data over the network. Note: This is only for Connections, so no UDP data transmission with this one.
ssize_t recv(int sockfd, void *buf, size_t len, int flags);	Receives data. Opposite of send, this call blocks until it reads the number of bytes specified.
int close(int fd);	Closes a file descriptor. Since this closes a file descriptor, it works on other things too! (Like text files)
int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);	Makes the specified port belong to you. The server is the only one to bind.
int listen(int sockfd, int backlog);	Waits for an incoming connection request. Blocking.
int accept(int sockfd, struct sockaddr *addr, socklen_t* addrlen);	Accepts a connection request. Important to note that this returns a new file descriptor to a new socket.

Read the man pages for more information. Google is your friend.

Useful Memory Management Functions

void* malloc(size_t size);	This is just like new only a little bit more complex. You will need to cast the data that malloc returns to the type you need.
void free(void *ptr);	Same as delete.
void *memset(void *str, int c, size_t n);	Sets the value of the memory you give it to the value specified. Typically used to set a chunk of memory to 0.
void *memcpy(void *str1, const void *str2, size_t n);	Self explanatory.
void *memmove(void *str1, const void *str2, size_t n);	Self explanatory.
sizeof(thing)	sizeof will correctly get you the size of any type of data you give it. sizeof DOES NOT give you the size of an array as it only works with types. Fun Fact: sizeof isnt a function, its an operator!

Bit Bashing

Definitions
& – Bitwise AND
| – Bitwise OR
~ – Bitwise NOT
^ – XOR
<< – Left Shift
>> – Right Shift
Logic Tables:
& AND
----0 1
0 | 0 0
1 | 0 1
| OR
----0 1
0 | 0 1
1 | 1 1
^ XOR
----0 1
0 | 0 1
1 | 1 0
~ NOT
0 becomes 1
1 becomes 0
<< Shift Left
Moves ALL bits left n spaces
0001 << 2 = 0100
Shift Right
0101 >> 2 = 0001
NOTE: Bits that are shifted too far (Like the example above) are gone so if you did
x = 0101 >> 2
x = x << 2
the answer would be 0100
So for:
  x = 00101000 (40)
  y = 01010000 (80)
x&y = 00000000= 0 (decimal)
x|y = 01111000 = 120 (decimal)
~x = 11010111 = -41 (decimal)
x^y = 01111000= 120 (decimal)
x << 1 = 01010000 = 80 (decimal)
x >> 1 = 00010100 = 20 (decimal)

More information found here
https://en.wikipedia.org/wiki/Bitwise_operations_in_C

Useful Linux Commands/Programs

grep	This is a REALLY useful one to search anything.https://www.gnu.org/savannah-checkouts/gnu/grep/manual/grep.html Important to note that you can pipe something into grep to search the output of another program.
valgrind	The queen of memory leak detection. If you get a segfault then this is your best friend.http://valgrind.org
gdb	Your debugger, this is a pretty big topic.https://www.gnu.org/software/gdb/
make	Run a makefile. A useful makefile tutorial can be found here:http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/

Basic commands like ls, pwd, and clear are not included here.