t

Command Line

Command	Options	Explanation
grep	print lines matching a pattern (grep <string> <filename>)	
	-v	invert-match: displays lines not containing the string
	-o	only show part of line that matches pattern
	-i	case insensitive
	-R	search all directories
	$	e.g. fish$ (looks for the word fish as the last word of a line)
	^	e.g. ^fish (looks for the word fish as the first word of a line)
	.	any character in regex except newlines
	*	0 or more of previous expression in regex
gcc	-o	creates an executable file from a c file, the order needs to be: gcc -o <executable-name> <name-of-file>
	-c	compiles a c file and creates an .o file of the same name
	-g	include debugging symbols
ls	list directory contents	
	-l	use a long listing format
	-a	shows all hidden directories, do not ignore entries starting with .
	-d	list directories themselves not their content
	-i	print the index number of each file
ps	process status	
	-e	show all processes -> to see every process on the system using standard syntax
	-f	do full format listing, adds additional columns
sort	write sorted concatenation of all files(s) to stdout (sort <option> <filename>)	
	-r	reverse result of comparisons
	-k	sort via key
uniq	report or omit repeated lines	
	-c	count number of occurrences
cat	concatenate files to print on stdout	
head	output first part of file to stdout; first 10 lines if not specified	
	-n	output first, N, number of lines (with leading ‘-’ print all but last N)
	-c	print first N bytes of each file (leading ‘-’ as above)
	-q	quiet, never print headers giving file name
tail	display last part of fi le to stdout; last 10 lines if not specified	
	-n	output last, N, number of lines

cut	remove sections from each line of files	
	-f	select only these fields; print any line that contains no delimiter character, unless -s is specified Can specify multiple fields with a comma (e.g. -f1,3,4) Can specify ranges with dashes. `N-` *N^{th} onwards*, `-M` *up to M*, `N-M` *n to m*
	-s	do not print lines not containing delimiters
	-d	specify delimiter (e.g. -d':')
wc	print newline, word and byte counts for each file	
	-l	number of newlines
	-c	byte count
	-m	character count
diff	compare files line by line	
	-q	report only when files differ
	-s	report identical files
svn	subversion command line client tool	
	commit	send changes from working copy to repository
	add	put files in directory under version control. Added to repository in next commit
	remove	remove files and directories from VC. Scheduled for deletion upon next commit and removed from working copy.
	move	move and/or name something in working copy or repository
	update	bring changes from repository into working copy
	info	display information about a local or remote item
	log	show log messages for a set revision(s) and/or path(s)
	status	print the status of working copy and files and directories
	diff	display differences between two revisions or paths
	revert	Undo all local edits. Svn revert <target>
chmod	change file modes i.e. change access controls (can't use if not owner)	
	u/g/o/a +/- r/w/x	user/group/others/all add/remove read/write/execute
	-c	only report when a change is made
	-R	change files and directories recursively
	-v	output a diagnostic for every file processed
	-f	suppress most error messages
ls	list directory contents	
	-l	lists files and read write execute info for each user group and other

	-i	list index number of each file
	-s	print allocated size of each file of each file
	-a	print all files including hidden ones (prefaced with '.')
rm	-r	remove directories and their contents recursively i.e. delete everything in the specified subdirectories
	-f	force -> ignore nonexistent files and arguments, never prompt
	-d	remove empty directories
	-v	explain what is being done
	*	remove all
mkdir	make directories (if they do not exist)	
	-m	set file mode like chmod
	-p	no error if existing parent, make parent directories as needed
rmdir	remove empty directories (if they are empty)	
	-p	remove directory and its ancestors e.g. 'rmdir a/b/c' i.e. 'rmdir a/b/c a/b a'
cp	copy files (cp <old> <new>)	
	-r	copy directories recursively
scp	secure copy: (remote file copy) files between host and network	
	-c	selects the cipher to use for encrypting data, this option is directly passed to ssh(1)
mv	move (**rename**) file; rename source to destination OR move source to directory	
	-f	force; do not prompt before overwriting
	-i	interactive; prompt before overwrite
vim	Vi IMproved, programmers text editor	
pico	simple text editor in style of Alpine Composer	
less	allows backward and forward movement in a file using arrow keys (i.e. cat with up and down, cat only has down)	
ln	makes links between files -s for symbolic -p for physical/hard link	

`*`	Matches any string, of any length
`foo*`	Matches any string beginning with `foo`
`*x*`	Matches any string containing an `x` (beginning, middle or end)
`*.tar.gz`	Matches any string ending with `.tar.gz`
`*.[ch]`	Matches any string ending with `.c` or `.h`
`foo?`	Matches `foot` or `foo$` but not `fools`

Command Line Examples

Select the **first and third column delimited by ‘ ‘ and sort by the first**

cat file.txt | cut -d’ ‘ -f1,3 | sort -k1

Get **only** the **12th line** of a given file

cat file.txt | head -12 | tail -1

Get **each line with cat and dog** in a given file

cat file.txt | grep “cat” | grep “dog”

Find **each line** which contains “CSSE1001” and does **not contain the word** “boring”

cat file.txt | grep “CSSE1001” | grep -v “boring”

Find the **number of times** “rowing” occurs **in a given file**

cat ainsley.txt | grep -o “rowing” | wc -l

For **all files** f1,f2,f3 show **all lines containing** “song”, “river” and “terrible”

cat f1 f2 f3 | grep song | grep river | grep terrible

For **all files** g1,g2,g3 show **all lines not containing** “song”, “river” and “awful”

cat g1 g2 g3 | grep -v song | grep -v river | grep -v awful

OR grep -ve grep -ve song -ve awful f1 f2 f3

Find **all lines** in file1 containing the word “dinosaur” and **store in file** called london

cat file1 | grep “dinosaur” > london OR grep dinosaur file1 > london

Show **all lines** in file1 **starting with** W

grep ^W file1

Show **all lines** in file2 **ending with** S

grep S$ file2

Modify path

export PATH = $PATH:newpath

Show second last line of file

cat file | tail -n 2 | head -n 1

Show fifth line of file

cat file | head -n 5 | tail -n 1

Show the third line and later (hide the first two lines)

cat file | tail -n +3

Show all but the last 10 lines (hide the last 10 lines)

cat file | head -n -10

File Permissions and Terminology

UNIX files user info

Example (ls -al)

drwxrwxrwx <number of links> <owner name> <owner group> <file size> <date/time last edited> <name>

Permissions

1st char	‘-’ is regular file, ‘d’ is directory, ‘l’ is link
r	read permission
w	write permission
x	execute permission
-	normal file, or does not have that permission
1st “rwx” triplet	owner permissions
2nd “rwx” triplet	group permissions (owner is member of group)
3rd “rwx” triplet	other permissions (users not in group)

#	Permission	rwx
7	read, write and execute	rwx
6	read and write	rw-
5	read and execute	r-x
4	read only	r--
3	write and execute	-wx
2	write only	-w-
1	execute only	--x
0	none	---

Chmod commands

Add read+write permissions to user.

```
chmod u+rw file
```

Take write+execute permissions from group

```
chmod g-wx file
```

Set permissions to group and others for executing only

```
chmod go=x file
```

Deny writing permissions to everyone

```
chmod a-w file
```

Use numbers for permissions (rwx-w-r-x)

```
chmod 725 file
```

Subdirectories = number of links - 2

- directories permissions
  - read: can perform ls
  - write: can create files
  - execute: can access that directory
- file permissions
  - read: can view files
  - write: can write to file
  - execute: can run/execute program or script

Mounting

- Allows additional file system to be added at a particular entry point of the current file system. Allows you to have more than one file system.
- Takes on a tree like structure

Symbolic (soft) links: ln -s <existing> <desired link> **n.b. file does NOT have to exist**

- If the file being pointed to changes, the link will break
- If the softlink is renamed, this will cause issues (as they don't share an inode as an identifier)

Hard links: ln <existing> <desired link> **n.b. file MUST exist**

- share the same inode (column 1 in ls -li)
- **ln can fail** because:
  - inodes aren't shared between different file systems
  - incorrect permissions e.g. don't have permissions on that directory
  - something already exists with that name
  - some systems do not allow cyclic links of directories
  - cannot hard link files belonging to other users

File Systems

- inode
  - lots of overhead: not good for lots of very small files
  - array of block ptrs for every file
  - there is a file size limit
  - quicker for accessing parts of files: O(1) -> lots of seeks
- linked list
  - copes better with lots of large files
  - no file size limit
  - no external fragmentation
  - faster sequential access but overall slower reading: O(n) -> lots of seeks further in

Fragmentation

- Internal: space allocated but not used eg: 4KB blocks, 5KB file = 2 blocks, 3KB wasted.
- External: unallocated spaces: too small to be useful, or too spread out.

Linked Systems File Size

- **keep units consistent**
  - n KiB = n*1024^1 bytes = n*2^{10} bytes
  - n MiB = n*1024^2 bytes = n*2^{20} bytes
  - n GiB = n*1024^3 bytes = n*2^{30} bytes
- **number of ptrs in block** = block size / block pointers
- A = size of direct = (num of direct ptrs) * (block size)
- B = size of indirect = (num of indirect ptrs) * (number of ptrs in block) * (block size)
- C = size of double indirect = (num of double indirect ptrs) * (number of ptrs in block)^2 * (block size)
- **total file system size** = A + B + C
- **number of blocks required to store files** = 1 for inode + number of direct blocks required
  - if size > A, then add number of blocks required in B + 1
  - if size > (A + B), then number of blocks required in C + number of indirect blocks required + 1

Virtual Memory

There are four steps to converting a virtual address to a physical address:

1. Calculate the page (Note: integer division is used for this):

$$page = \frac{virtual\ address}{page\ size}$$

2. Calculate the page offset (Note: $\%$ is the modulus operator):

$$offset = virtual\ address\ \%\ page\ size$$

3. Find the frame by mapping the page to a frame via the page table
4. Calculate the physical address:

$$physical\ address = frame * page\ size + offset$$

It is not always possible to get a physical address from a virtual address. If the page table has a page listed as *invalid*, then a segmentation fault would occur, instead of accessing the physical address.

$$M\ level\ Page\ Table\ size = (n^{m-1} + m^{m-2} + ... + m^0) \times page\ size$$

$$Virtual\ Memory\ size\ of\ an\ M\ level\ page\ table = (n^m) \times page\ size$$

Causes of Segmentation Faults

- when you dereference NULL pointer
- trying to use/point to memory that has not been allocated or has already been freed
- reading or writing to memory you don't have access/write permissions to
- memory is owned by another process
- being a shitty programmer i.e. because you made it happen
- because quicksand

Causes of Page Faults

- doesn't exist in TLB (translation lookaside buffer) -> special fast-lookup cache for fast memory access
- when process/object is on disk but not in memory

C Standards

General Comments/Notes

- the C system does not control how memory is used
- does NOT supported nested functions
- avoid global variables except for signal handlers
- **integer math is closed**
  - `int`/`int` = `int` however `float`/`int` = `float`
- no function reloading
- `sizeof(char)` = 1 (byte); `sizeof(int)` = 4 (bytes) usually
- reads from right to left
  - `a = b = c = 0`
- passed by value
- c files compile; o files link
- 0 is false, anything else is true

ANSI i.e. C90

- must declare variables before statements
- can initialise inside the braces of a for or while loop, but cannot declare.
- boolean is NOT a type, but can be typedef-ed
- no single-line comments allowed, only block comments

C99

- booleans is a valid type "`bool`" #include <stdbool.h>

Flags

- -**Werror**: make all warnings into errors
- -**pedantic**: reject all programs that use forbidden extensions
- -**Wall**: all warnings which disgruntled 2310 students consider questionable
- -**Wextra**: extra about constructions that some users consider questionable

Networks and Subnets

- **network address** = IP addr & subnet mask *(lowest address in network)*
- **host address** = IP addr & -(subnet mask)
- **broadcast address** = IP addr | -(subnet mask) *(highest address in network)*

All Possible Subnet Masks and Example

Mask	bit conversion	# networks	# hosts
255	1111 1111	256	1
254	1111 1110	128	2
252	1111 1100	64	4
248	1111 1000	32	8
240	1111 0000	16	16
224	1110 0000	8	32
196	1100 0000	4	64
128	1000 0000	2	128
0	0000 0000	1	256

Find the smallest network which accomodates all these addresses. NOTE: The router's address is also part of the network

```
XNOR ip addresses
12.16.1.8  -> 00001100 00010000 00000001 00001000
12.16.1.9  -> 00001100 00010000 00000001 00001001
12.16.1.10 -> 00001100 00010000 00000001 00001010
12.16.1.11 -> 00001100 00010000 00000001 00001011
12.16.0.7  -> 00001100 00010000 00000000 00000111
network addr: ======== ======== =======
              11111111 11111111 11111110 00000000
              255      255      254      0

Thus the netmask is 255.255.254.0
```

To find the network address of the smallest network:

```
AND ip addresses
12.16.1.8  -> 00001100 00010000 00000001 00001000
12.16.1.9  -> 00001100 00010000 00000001 00001001
12.16.1.10 -> 00001100 00010000 00000001 00001010
12.16.1.11 -> 00001100 00010000 00000001 00001011
12.16.0.7  -> 00001100 00010000 00000000 00000111
network addr: ======== ======== =======
              00001100 00010000 00000000 00000000
              12       16       0        0

Thus the netmask is 12.16.0.0
```

Note: "usable" # of hosts is 2 less due to network and broadcast addresses

IP stack

Layer	Examples
Application (5)	SSH, HTTP, bittorrent, skype, FTP, Telnet (uses TCP), netcat (uses TCP & UDP), URL **socket** = IP address + port
Transport (4)	**TCP** - transmission control protocol delivery is reliable and in order **UDP** - datagram protocol connectionless and unreliable **port** and **port numbers**: one interface has 65535 ports but a single IP
Network (3)	communicating with any host on the internet **two tasks**: 1) find node close to destination 2) send packet in that direction (**routing**) protocol = **IP address** n.b. different link layers have different rules ethernet interface needs MAC address + >= 1 IP address
Link (2)	talking to nodes without intermediary e.g. ethernet, wifi, infrared, carrier pigeon need **MAC** (medium access control) **addresses** **node**: can have a number or link layer interfaces but you don't always need a node 127.0.0.1 = loopback address
Physical (1)	medium through which signal travels e.g. air & EM waves, voltage & wire

Network Address Translation (NAT)

- a way of dealing with private IP addresses so the internet does not run out of IVP4 addresses
- when num of IP's assigned to you is less than total num of computers trying to access internet, assigns entity/organisation in a single IP
- some IP's are non-routable, this is how they access the internet
- example: only occurs at boundary of UQ to interact with external networks
- n.b. every interface has it's own IP address

Gateway Address

- idea: there is a local network and everybody else, to talk to everybody else must use a gateway
- gateway: usually 1:1
- router: usually 1:N

Domain Name System (DNS)

- hierarchical naming system that maps names to IP addresses
    - this is how you get to google.com etc

File Handling Functions

function	return value	
	on success	**on failure**
fopen	file pointer	NULL pointer
fclose	0	EOF
fflush	0	EOF
scanf	number of input items successfully matched	EOF (i.e. -1)
fgetc, getc, getchar	unsigned char cast to an int (i.e. not a character)	EOF (i.e. -1)
gets, fgets	string read	NULL on error or when EOF occurs
ungetc	character	EOF (i.e. -1)
fputc, putc, putchar	unsigned char cast to an int (i.e. not a character)	EOF
puts, fputs	non-negative number	EOF
fprintf	number of characters printed excluding '\0'	negative value returned

String Handling

#include <string.h>

Function and Description	Return Value
int **strcmp**(const char *s1, const char *s2); compares the two strings s1 and s2 int **strncmp**(const char *s1, const char *s2, size_t n); compares the only first (at most) n bytes	int < 0 if s1 < s2 int = 0 if s1 = s2 int > 0 if s1 > s2
char ***strcpy**(char *dest, const char *src); copies the string pointed to by src, including the terminating null byte ('\0'), to the buffer pointed to by dest char ***strncpy**(char *dest, const char *src, size_t n); at most n bytes are copied	a pointer to the destination string dest
char ***strcat**(char *dest, const char *src); appends the src string to the dest string, overwriting the terminating null byte ('\0') at the end of dest, and then adds a terminating null byte char ***strncat**(char *dest, const char *src, size_t n); If src contains n or more bytes, strncat() writes n+1 bytes to dest (n from src plus the terminating null byte). Therefore, the size of dest must be at least strlen(dest)+n+1	a pointer to the resulting string dest
char ***strtok**(char *str, const char *delim); breaks a string into a sequence of zero or more nonempty tokens char ***strtok_r**(char *str, const char *delim, char **saveptr); thread-safe version. doesn't use static char* pointer.	return a pointer to the next token, or NULL if there are no more tokens
char ***strstr**(const char *haystack, const char *needle); finds the first occurrence of the substring needle in the string haystack. The terminating null bytes ('\0') are not compared	a pointer to the beginning of the substring, or NULL if the substring is not found

Processes

Function	Return Value
int **pipe**(int pipefd[2]);	on success: 0 on error: -1
pid_t **fork**();	on success: child_pid to parent, 0 to child on error: -1
int **dup2**(int oldfd, int newfd);	on success: new fd on error: -1
int **execl**(const char *path, const char *arg, …, NULL); int **execlp**(const char *file, const char *arg, …, NULL); int **execv**(const char *path, char *const argv[]); int **execvp**(const char *file, char *const argv[]);	only returns on error: -1
pid_t **wait**(int *status); is **blocking call** pid_t **waitpid**(pid_t pid, int *status, int options);	on success: child_pid on error: -1 (or if you have no children)

Flags

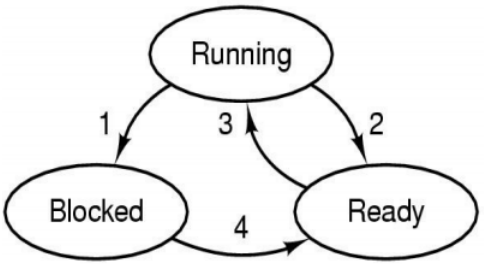
- **WNOHANG**: to test if child process has terminated, returns immediately if no child has exited
- **WEXITSTATUS**(status) returns process exit status
- **WIFEXITED**(status) returns true if process exited normally
- **WIFSIGNALED**(status): returns true if child was terminated by signal

Scheduler

- **long term** (job scheduler): selects which processes which should be brought back into the ready queue
- **short term** (CPU scheduler): selects which process should be executed next and allocates CPU

Process Possible States

- running
- blocked
- ready



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

Threads and Thread Safety

#include <pthread.h> & -pthread

Function name and description	Return Value
pthread_t threadID threadID type	n.a.
int **pthread_create**(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg);	on success 0 on error - error num and contents of thread are undefined
int **pthread_join**(pthread_t thread, void **retval); waits for the thread specified by thread to terminate. If that thread has already terminated, then pthread_join() returns immediately. The thread specified by thread must be joinable	on success 0 on error - error number

int **pthread_detach**(pthread_t thread); marks the thread identified by thread as detached. When a detached thread terminates, its resources are automatically released back to the system without the need for another thread to join with the terminated thread (function called on a thread so it doesn't need to be joined)	on success 0 or error - error number
void **pthread_exit**(void *retval); terminates the calling thread and returns a value via retval that (if the thread is joinable) is available to another thread in the same process that calls pthread_join(3)	does not return to caller
pthread_t **pthread_self**(void); returns the ID of the calling thread, same value as &thread in pthread_create()	calling thread's ID
int **pthread_cancel**(pthread_t thread); sends a cancellation request to the thread thread. Whether and when the target thread reacts to the cancellation request depends on two attributes that are under the control of that thread: it's cancelability state and type	on success 0 on error - error number

Threads Possible States

- ready
- running
- blocked
- terminated -> recycling

What data is shared?

- Global variables: one copy per process
- Local variables: one copy per thread i.e. registers and stack
- Static variables: one copy per process

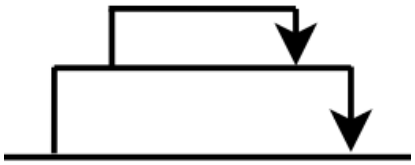
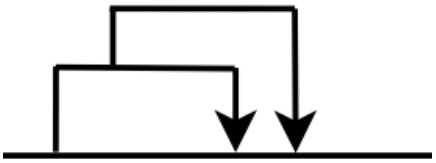
Semaphores

- sem_init(&lock, 0, value); 0 = not shared between processes, value = num of semaphores
- sem_wait(&lock); - locks/grabs until value < 0
- sem_post(&lock); - unlocks/releases

Mutexes

- pthread_mutex_init(&lock, NULL);
- pthread_mutex_unlock(&lock);
- pthread_mutex_lock(&lock);

Process and Thread Diagrams

Diagram	Thread or Process?
	This can be either one. Threads can be reaped by any other thread. Processes can only be reaped by parent.
	This can also be either one. Even though the parent terminates early, the child is then adopted by init and then reaped. Both threads and processes can be reaped by init.

Signals

- SIGKILL (kill -9 pid) **cannot fail**
- **signals can fail** because:
  - pid does not exist
  - user does not have correct permissions
- SIGHUP (1): terminates -> hang up
- SIGTERM (15): terminates -> shut down cleanly
- SIGPIPE: terminates -> write on a pipe with no one to read it

ID	Name	Default Action	Corresponding Event
2	SIGINT	Terminate	Interrupt from keyboard (Ctrl-c)
9	SIGKILL	Terminate	Kill program (cannot override or ignore)
11	SIGSEGV	Terminate	Segmentation violation
14	SIGALRM	Terminate	Timer signal
17	SIGCHLD	Ignore	Child stopped or terminated

Signal Handlers

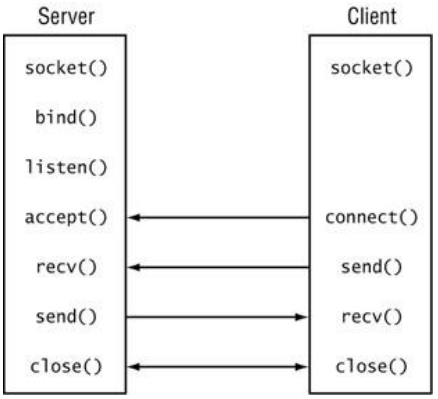
sigaction(int signalNum, struct sigaction* act, struct sigaction* oldact)

struct **sigaction** {

    void (*sa_handler)(int);

    int sa_flags;

}



Socket System Calls

#include <sys/types.h>, #include <sys/socket.h>

- why **htons**(), **ntohs**(), **htonl**(), **ntohl**():
  - because endianness of systems doesn't always match network endianness (big endian)

Function	Return Value
int **accept**(int sockfd, struct sockaddr *addr, socklen_t *addrlen); accepts pending connections for listening socket (sockfd) and creates new connected socket (not in listening state). is a **blocking call**	on success: non-neg file descriptor on error: -1
int **socket**(int domain, int type, int protocol); creates a socket as a new communication point	
int **connect**(int sockfd, const struct sockaddr *addr, socklen_t addrlen); attempt to establish a connection to a socket given an fd to an address specify: **SOCK_STREAM** for TCP	on success: 0 on error: -1
int **bind**(int sockfd, const struct sockaddr *addr, socklen_t addrlen); after a socket is created, bind attaches a local address to it	
int **listen**(int sockfd, int backlog); marks a socket to be ready for listening, backlog is max length of pending queue	
int **getaddrinfo**(const char *node, const char *service, const struct addrinfo *hints, struct addrinfo **res); **network address translation** service, provide a node and interface (i.e. service)	on success: 0 on error: error code
int **getnameinfo**(const struct sockaddr *sa, socklen_t salen, char *host, size_t hostlen, char *serv, size_t servlen, int flags); converts socket to address to host and service (i.e. port and interface)	
int **getsockname**(int sockfd, struct sockaddr *addr, socklen_t *addrlen); returns address of socket	on success: 0 on error: -1

Operator Precedence

CSSE 2310 7231

Operator Precedence and Associativity

Operators		Associativity
( ) [ ] -> .	(evaluated first)	Left to right
! ~ + - ++ -- & *	(unary versions)	Right to left
* / %		Left to right
+ -		Left to right
<< >>		Left to right
< <= > >=		Left to right
== !=		Left to right
&	(bitwise and)	Left to right
^	(bitwise xor)	Left to right
|	(bitwise or)	Left to right
&&	(logical and)	Left to right
||	(logical or)	Left to right
?:		Right to left
= *= /= %= += -= &= ^= |= <<= >>=	(assignment)	Right to left
,	(evaluated last)	Left to right

ASCII TABLE

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Source