**UNX511/DPS912: Final Exam Review**

**Exclusions**: Kernel Device Drivers, Shared Libraries.

1. Tell what the following command-line commands do:
	1. [nm](http://man7.org/linux/man-pages/man1/nm.1.html), [ldd](http://man7.org/linux/man-pages/man1/ldd.1.html),
	2. [top](http://man7.org/linux/man-pages/man1/top.1.html),
	3. [netstat](http://man7.org/linux/man-pages/man8/netstat.8.html), [ss](http://man7.org/linux/man-pages/man8/ss.8.html),
	4. [pmap](http://man7.org/linux/man-pages/man1/pmap.1.html).
2. What do the following Linux functions do inside of C++ code:
	1. [perror()](http://man7.org/linux/man-pages/man3/perror.3.html) and [strerror()](http://man7.org/linux/man-pages/man3/strerror.3.html),
	2. [dup() and dup2()](http://man7.org/linux/man-pages/man2/dup.2.html),
	3. [fcntl()](http://man7.org/linux/man-pages/man2/fcntl.2.html),
	4. [pread() and pwrite()](http://man7.org/linux/man-pages/man2/pwrite.2.html),
	5. [readv() and writev()](http://man7.org/linux/man-pages/man2/readv.2.html),
	6. [ioctl()](http://man7.org/linux/man-pages/man2/ioctl.2.html),
	7. [fork()](http://man7.org/linux/man-pages/man2/fork.2.html), [vfork()](http://man7.org/linux/man-pages/man2/vfork.2.html), [exit()](http://man7.org/linux/man-pages/man3/exit.3.html), [atexit()](http://man7.org/linux/man-pages/man3/atexit.3.html), [on\_exit()](http://man7.org/linux/man-pages/man3/on_exit.3.html), [wait(), waitpid()](http://man7.org/linux/man-pages/man2/wait.2.html), and [execve()](http://man7.org/linux/man-pages/man3/exec.3.html),
	8. [kill()](http://man7.org/linux/man-pages/man2/kill.2.html),
	9. [signal()](http://man7.org/linux/man-pages/man2/signal.2.html),
	10. [socket()](http://man7.org/linux/man-pages/man2/socket.2.html), [bind()](http://man7.org/linux/man-pages/man2/bind.2.html), [listen()](http://man7.org/linux/man-pages/man2/listen.2.html), [connect()](http://man7.org/linux/man-pages/man2/connect.2.html), [accept()](http://man7.org/linux/man-pages/man2/accept.2.html), [select()/FD\_SET()/FD\_ISSET()](http://man7.org/linux/man-pages/man2/select.2.html), [inet\_pton()](http://man7.org/linux/man-pages/man3/inet_pton.3.html), [inet\_ntop()](http://man7.org/linux/man-pages/man3/inet_ntop.3.html), [read()](http://man7.org/linux/man-pages/man2/read.2.html), [write()](http://man7.org/linux/man-pages/man2/write.2.html), [send()/sendto()](http://man7.org/linux/man-pages/man2/send.2.html), [recv()/recvfrom()](http://man7.org/linux/man-pages/man2/recv.2.html), [close()](http://man7.org/linux/man-pages/man2/close.2.html), [unlink()](http://man7.org/linux/man-pages/man2/unlink.2.html),
	11. [pipe()](http://man7.org/linux/man-pages/man2/pipe.2.html), [mkfifo()](http://man7.org/linux/man-pages/man3/mkfifo.3.html),
	12. [pthread\_create()](http://man7.org/linux/man-pages/man3/pthread_create.3.html), [pthread\_join()](http://man7.org/linux/man-pages/man3/pthread_join.3.html), [pthread\_exit()](http://man7.org/linux/man-pages/man3/pthread_exit.3.html),
	13. [pthread\_mutex\_init(), pthread\_mutex\_destroy()](https://linux.die.net/man/3/pthread_mutex_init), [pthread\_mutex\_lock(), pthread\_mutex\_unlock()](https://linux.die.net/man/3/pthread_mutex_lock),
	14. [ftok()](http://man7.org/linux/man-pages/man3/ftok.3.html), [msgget()](http://man7.org/linux/man-pages/man2/msgget.2.html), [msgsnd(), msgrcv()](http://man7.org/linux/man-pages/man2/msgsnd.2.html), [msgctl()](http://man7.org/linux/man-pages/man2/msgctl.2.html),
	15. [sem\_init()](http://man7.org/linux/man-pages/man3/sem_init.3.html), [sem\_open()](http://man7.org/linux/man-pages/man3/sem_open.3.html), [sem\_wait()](http://man7.org/linux/man-pages/man3/sem_wait.3.html), [sem\_post()](http://man7.org/linux/man-pages/man3/sem_post.3.html), [sem\_destroy()](http://man7.org/linux/man-pages/man3/sem_destroy.3.html), [sem\_unlink()](http://man7.org/linux/man-pages/man3/sem_unlink.3.html),
	16. [shmget()](http://man7.org/linux/man-pages/man2/shmget.2.html), [shmat()](http://man7.org/linux/man-pages/man2/shmat.2.html), [shmdt()](https://linux.die.net/man/2/shmdt), [shmctl()](http://man7.org/linux/man-pages/man2/shmctl.2.html).
3. Identify all the special symbols inside the following Makefile. For answers, see [Makefile](https://github-pages.senecapolytechnic.ca/unx511/Week8/AdvancedMakefiles/Makefile4/Makefile).

CC=g++

CFLAGS=-I

CFLAGS+=-Wall

FILES=hellomake.c

FILES+=hellofunc.c

OBJ=hellomake.o

OBJ+=hellofunc.o

DEPS=hellomake.h

%.o: %.c $(DEPS)

 $(CC) -c -o $@ $< $(CFLAGS)

hellomake: $(OBJ)

 $(CC) $(CFLAGS) $^ -o $@

clean:

 rm -f \*.o hellomake

all: hellomake

1. What do the following signals do? For answers, see [List of Signals](https://github-pages.senecapolytechnic.ca/unx511/Week4/signals.pdf)

SIGHUP, SIGINT, SIGKILL, SIGPIPE, SIGALRM, SIGTSTP.

1. How do you send signals to processes from the command line? See [Sending signals to Processes](https://bash.cyberciti.biz/guide/Sending_signal_to_Processes).
2. What are the steps in setting up sockets between a client and server? See [C++ Tutorial: Sockets - Server and Client](https://www.bogotobogo.com/cplusplus/sockets_server_client.php).
3. Below is a summary of Inter-Process Communications within Linux:



* 1. Tell me two things about a [pipe](http://man7.org/linux/man-pages/man7/pipe.7.html).
	2. Tell me two things about a [fifo](http://man7.org/linux/man-pages/man7/fifo.7.html).
	3. Tell me two things about a [socket](http://man7.org/linux/man-pages/man7/socket.7.html).
	4. Tell me two things about a [message queue](http://man7.org/linux/man-pages/man7/mq_overview.7.html).
	5. Tell me two things about [shared memory](http://man7.org/linux/man-pages/man7/shm_overview.7.html).
1. Write code to send the alphabet (lower case) from a client to server using stream sockets. The server converts the alphabet to upper case and sends it back to the client. See [client.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/streamSockets/client.cpp), [server.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/streamSockets/server.cpp) and [Makefile](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/streamSockets/Makefile).
2. Write code to send the alphabet (lower case) from a client to server using datagram sockets. The server converts the alphabet to upper case and sends it back to the client. See [msgClient.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/datagramSockets/msgClient.cpp), [msgServer.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/datagramSockets/msgServer.cpp) and [Makefile](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/datagramSockets/Makefile).
3. Write code to send the alphabet (lower case) from a client to server using fifo’s. The server converts the alphabet to upper case and sends it back to the client. See [fifo1.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/fifo/fifo1.cpp), [fifo2.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/fifo/fifo2.cpp) and [Makefile](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/fifo/Makefile).
4. Write code to send the alphabet (lower case) from a parent to a child using pipes. The server converts the alphabet to upper case and sends it back to the client. See [pipe.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/pipe/pipe.cpp) and [Makefile](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/pipe/Makefile).
5. What does the following code do?

// rock.c - C program to illustrate execlp()

//

// 04-Apr-19 M. Watler Created.

//

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main()

{

 int pid;

 printf("\nWe are the champions!\n");

 pid = fork();

 if( pid > 0 ){

 printf("We will\n");

 sleep(1);

 execlp("./rock", "rock", NULL);

 printf("rock you!\n");

 }else if( pid == 0 ){

 printf("I've paid my debts\n");

 execlp("/bin/date", "date", NULL);

 printf("I will never lose!\n");

 }

 printf("The Leafs are the champions! (in 1967)\n");

}

See [rock.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/rock/rock.cpp) and [Makefile](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/rock/Makefile). See [sampleOutput.txt](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/rock/sampleOutput.txt) for sample output of the above code.

1. What does the following code do?

// pipeline.cpp - C program to illustrate piping.

//

// 04-Apr-19 M. Watler Created.

//

#include <stdlib.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

char \*cmd1[] = { "du", "-a", 0 };

char \*cmd2[] = { "wc", 0 };

int main(int argc, char \*\*argv) {

 int pid, status;

 int fd[2];

 pipe(fd);

 switch (pid = fork()) {

 case 0: /\* child \*/

 int pid2;

 switch (pid2 = fork()) {

 case 0: /\* child \*/

 dup2(fd[0], 0);

 close(fd[1]);//child doesn’t need this end of the pipe

 execvp(cmd2[0], cmd2);

 perror(cmd2[0]);

 default: /\* parent \*/

 dup2(fd[1], 1);

 close(fd[0]);//parent doesn’t need this end of the pipe

 execvp(cmd1[0], cmd1);

 perror(cmd1[0]);

 case -1:

 perror("fork");

 exit(1);

 }

 exit(0);

 default: /\* parent \*/

 while ((pid = wait(&status)) != -1)

 fprintf(stderr, "process %d exits with %d\n", pid,

 WEXITSTATUS(status));

 break;

 case -1:

 perror("fork");

 exit(1);

 }

 exit(0);

}

See [pipeline.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/pipeline/pipeline.cpp) and [Makefile](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/pipeline/Makefile). See [sampleOutput.txt](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/pipeline/sampleOutput.txt) for sample output of the above code.

1. Write code to send the following audio wave hexadecimal data from client to server using shared memory. You are given the following:

int audioWav[]={

 0x5249, 0x4646, 0x3a60, 0x1000, 0x5741, 0x5645, 0x666d, 0x7420,

 0x1000, 0x0000, 0x0100, 0x0200, 0x401f, 0x0000, 0x007d, 0x0000,

 0x0400, 0x1000, 0x6461, 0x7461, 0x345f, 0x1000, 0x8eff, 0x1700,

 0x0fff, 0x0300, 0xe3fe, 0xe3ff, 0x93ff, 0x2c00, 0xceff, 0x0c00

};

See [client.h](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/sharedMemory/client.h), [client1.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/sharedMemory/client1.cpp), [client2.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/sharedMemory/client2.cpp) and [Makefile](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/sharedMemory/Makefile).

1. Write a general client-server application using stream internet sockets that transmits keyboard input from client to the server with the server responding “Received” every time it receives a message from the client.

The IP address is local and the port number is passed by command line argument.

Both the client and server should receive messages inside a receive thread, and send messages inside a send thread.

Be sure to include any mutexes wherever required.

Have the code perform a controlled shutdown on ctrl-C.

Be sure to use blocking sockets.

If the client sends “Quit”, shut everything down on the server and client.

See [client.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/threads/client.cpp), [server.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/threads/server.cpp) and [Makefile](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/threads/Makefile).

1. Write three clients that attempt to open a file “dentists.txt” and read from it. Only one of the clients can open the file at any one time. Their access to this shared resource must be synchronized. Each client should read 16 bytes at a time, then sleep for 1 second. When a client has read 3000 bytes, it should start reading from the beginning of the file again. The clients should start up one after the other. All clients should wait until the last client starts up. The clients should perform a controlled shutdown on ctrl-C. The file can be found at: [dentists.txt](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/semaphore/dentists.txt).

See [client.h](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/semaphore/client.h), [client1.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/semaphore/client1.cpp), [client2.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/semaphore/client2.cpp), [client3.cpp](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/semaphore/client3.cpp) and [Makefile](https://github-pages.senecapolytechnic.ca/unx511/FinalReview/semaphore/Makefile).