**Unx511/DPS912: Mid-Term Review**

1. **The History of Linux** (for answers see [IntroToLinux](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week1/IntroToLinux.docx))
   1. Linux has its roots in which two operating systems?
   2. Is Linux a UNIX-derivative?
   3. How similar is Linux to UNIX?
   4. What is the Linux kernel?
   5. What is a Linux distribution?
   6. How is Linux superior to Windows?
   7. How is Windows superior to Linux?
2. **The Linux File-System** (for answers see [LinuxFilesystem](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week1/LinuxFilesystem.docx))
   1. What do you expect to see under the /bin directory?
   2. What do you expect to see under the /dev directory?
   3. What do you expect to see under the /home directory?
   4. What do you expect to see under the /lib directory?
   5. What do you expect to see under the /proc directory?
   6. What do you expect to see under the /var directory?
   7. What do you expect to see under the /bin directory?
3. **Some common Linux commands**
   1. What does nm do? (See [nm](https://linux.die.net/man/1/nm))
   2. What does ldd do? (See [ldd](http://man7.org/linux/man-pages/man1/ldd.1.html))
   3. What does top do? (See [top](https://linux.die.net/man/1/top))
   4. What does netstat do? (See [netstat](https://linux.die.net/man/8/netstat))
   5. What does tail -f do? (See [tail](http://man7.org/linux/man-pages/man1/tail.1.html))
4. **Makefiles**
   1. How do you create a Makefile for building an executable from the files Math.cpp, General.cpp, Geometry.cpp, and Conversion.cpp? (For the answer, see [Makefile](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week1/Makefile))
   2. How do you create a Makefile for building a static library libMath.a from the files General.cpp, Geometry.cpp and Conversions.cpp? (For the answer, see [Makefile](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week1/MathLib/Makefile))
   3. How do you create a Makefile that links in the static library libMath.a with its header file to build MathTest with the file MathTest.cpp? (For the answer, see [Makefile](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week1/MathTest/Makefile))
5. **Error Reporting**
   1. The file [ErrorMsg.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week1/ErrorMsg/ErrorMsg.cpp) demonstrates perror and strerror. What is the difference between perror and strerror? (See [perror](http://man7.org/linux/man-pages/man3/perror.3.html) and [strerror](http://man7.org/linux/man-pages/man3/strerror.3.html)).
   2. In the program [MiniMath.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week1/ErrorMsg/MiniMath.cpp), what does the following print out?

**cout << ErrMsg[Err\_DivideByZero] << endl << endl;**//Answer: “Divide by Zero”

1. **File Input/Output**
   1. Write a program that copies one file to another 16 bytes at a time using Linux system functions. (For the answer, see [SimpleFile.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week2/SimpleFile/SimpleFile.cpp))
   2. Do the same as in part 5a but start 100 bytes into the input file. (For the answer, see [SeekFile.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week2/SimpleFile/SeekFile.cpp))
   3. Which is faster, Linux’s open/read/write/close or the C-library’s fopen/fscanf/fprintf/fclose? (Answer: it depends. The C-library buffers its reads and writes. If Linux reads and writes a large number of bytes at a time, Linux will be faster. If Linux reads and writes one byte at a time, the C-library will be faster)
2. **IOCTL’s – Input/Output Control**
   1. How can a program running in user-space communicate with the kernel space? (For the answer, see [Controlling Hardware with ioctls](https://www.linuxjournal.com/article/6908))
   2. Write code that opens the device driver files /dev/sda and /dev/sda1, retrieves the disk parameters from each (HDIO\_GETGEO) and retrieves the device size from each (BLKGETSIZE). For the disk parameters, print out the heads, sectors, cylinders, and start. For the device size, print out the block size. (For the answer, see [diskDrive.cpp](https://ict.senecacollege.ca/~miguel.watler/courses/unx511/Week2/diskDrive/diskDrive.cpp))
3. **File Descriptors**
   1. Write a program that redirects the standard error output channel to the file Error.log. (For the answer see [fileDup.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week3/dup/fileDup.cpp))
   2. Which commands read from a file and writes to a file descriptor at a given offset? (Answer: see [pread and pwrite](http://man7.org/linux/man-pages/man2/pwrite.2.html) and [offset.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week3/offset/offset.cpp))
   3. Which commands can read or write an entire data structure to a file descriptor? (Answer: see [readv and writev](http://man7.org/linux/man-pages/man2/readv.2.html) and [car.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week3/car/car.cpp))
4. **Child Creation (fork, exec)**
   1. Describe the six memory segments of a process. (For the answer see [ProcessAddressSpace](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/ProcessAddressSpace.docx)).
   2. What command reports the memory map of a process? (Answer: [pmap](https://linux.die.net/man/1/pmap))
   3. What does fork() do as compared with vfork()? (See [ProcessCreationAndTermination](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/ProcessCreationAndTermination.docx))
   4. What does exit() do as compared with on\_exit()? (See [ProcessCreationAndTermination](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/ProcessCreationAndTermination.docx))
   5. What does wait() do as compared with waitpid()? (See [ProcessCreationAndTermination](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/ProcessCreationAndTermination.docx))
   6. What is the purpose of the exec() family of functions?

(See [ProcessCreationAndTermination](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/ProcessCreationAndTermination.docx))

* 1. Write a simple program that creates a child. Both parent and child count from 0 to 9 seconds printing out their respective pid’s. (See [fork1.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/fork/fork1.cpp))
  2. Write a simple program that creates four children. Each child counts from 0 to 9. As each child terminates, the parent reports the pid of the child that terminates.

(See [fork2.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/fork/fork2.cpp))

* 1. Write a program that forks a child which execs “ls -al”. (See [exec1.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/exec/exec1.cpp))
  2. Explain how on\_exit is used to clean up code in the files [openTest.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/exit/openTest.cpp) and [fopenTest.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/exit/fopenTest.cpp).

1. **Software Interrupts (signals)**
   1. Explain what happens when a software interrupt is generated. What happens to the instructions that are being executed, where does the program counter go, when does it return. (For the answer, see [software\_interrupt](http://www.linfo.org/software_interrupt.html))

FYI – A program counter is a register in a computer processor that contains the address (location) of the instruction being executed at the current time. As each instruction gets fetched, the program counter increases its stored value by 1.

* 1. Explain the signals SIGINT, SIGKILL, SIGUSR1, SIGUSR2, SIGTSTP. (See [signals](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/signals.pdf))
  2. Which signal is generated by a ctrl-C? (Answer: SIGINT)
  3. Which signal is generated by a ctrl-Z? (Answer: SIGTSTP)
  4. Which signal will violently kill your program? (Answer: SIGKILL)
  5. Which command from the command line can be used to send a signal to a process? (Answer: [kill](http://man7.org/linux/man-pages/man2/kill.2.html))
  6. What do the following commands from the command line do? (For the answer, see [Sending\_signal\_to\_Processes](https://bash.cyberciti.biz/guide/Sending_signal_to_Processes))

**kill -9 1234**

**kill -KILL 1234**

**kill -SIGKILL 1234**

* 1. How can you register a signal handler in your program? (Answer: see [signal](http://man7.org/linux/man-pages/man2/signal.2.html) or [sigaction](http://man7.org/linux/man-pages/man2/sigaction.2.html))
  2. Write a program that handles the following signals: SIGINT, SIGUSR1, SIGUSR2 using signal. (For the answer see [recvsig2.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/sigint/recvsig2.cpp))
  3. Write a program that spawns 4 children, then shuts them down one by one with SIGINT (ctrl-C). (For the answer see [sigact.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/sigaction/sigact.cpp))
  4. Write a program that cannot be interrupted with a ctrl-C or ctrl-Z. (for the answer see [uninterrupt.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/uninterrupt/uninterrupt.cpp))
  5. Write a program that forks and execs 2 child processes and synchronizes them via a signal. (See [sysmonExec.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/exec/sysmonExec.cpp) and [intfMonitor.cpp](https://scs.senecac.on.ca/~miguel.watler/courses/unx511/Week4/exec/intfMonitor.cpp)) Try the same with sigaction.