```
/* CPSC 457 (Winter 2019)
 * Week 2 - Section 2
 * Sina Keshvadi
 *
 * Notes: No error handling!
 */
```

System calls - System calls provide an interface to the services made available by an operating system.
Mostly, developers are not aware of the details of system calls, they use APIs(Application Programming Interface)

API - A set of functions that are available to an application programmer with:
◦ Function names
◦ Parameter types
◦ Return values

System call examples:

Type	Windows	Unix
Process Control	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File Manipulation	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Information Maintenance	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()

fork()

- fork() duplicates the current process
- The only way to differentiate the child and parent process is looking to the return value of the function
    - Returns 0 in the child process
    - Returns child process pid in the parent
- Both parent and child continue execution after fork function call
- fork() is the only way to create a process in Unix-like operating systems

fork bomb

Duplicates the process infinitely many times, crashes the system due to resource starvation

```
#include <stdio.h>
#include <unistd.h>

int main() {
   while(1){
      fork();
   }
   return 0;
}
```

Processes in Linux

View running processes
◦ ps command (with multiple options)
 Gives a snapshot of running processes

◦ top or htop command
 Continuous statistics until the user types q

Kill a running process
◦ Get the process ID using ps

kill pid or kill -9 pid
==========
Some UNIX utilities
head
sort
find
awk
==========
head

Print the first N number of lines
> head -n 5 countries.txt

Print all but not the last N lines
> head -n -5 countries.txt

You may pass the output of other commands to the head command via pipe | as shown below:
> ls | head -n 3

==========
Sort

sort simply sorts the file in alphabetical order:
> sort countries.txt

sort removes the duplicates using the -u option:
> sort -u numbers.txt

To sort a file numericallly:
> sort -n numbers.txt

sort file numerically in reverse order:
> sort -nr numbers.txt

sort can sort multiple files as well.
> sort -n countries.txt numbers.txt

Sort, merge and remove duplicates:
> sort -nu countries.txt numbers.txt

sorting a file containing multiple fields (the file got sorted on the 1st field, by default):
> sort mix.txt

sort file on the basis of 1st field:
> sort -t"," -k1,1 mix.txt

sorting file on the basis of the 2nd field:
> sort -t"," -k2,2 mix.txt

This is being more explicit. '-t' option is used to provide the delimiter in case of files with delimiter. '-k' is used to specify the keys on the basis of which the sorting has to be done. The format of '-k' is : '-km,n' where m is the starting key and n is the ending key. In other words, sort can be used to sort on a range of fields just like how the group by in sql does. In our case, since the sorting is on the 1st field alone, we speciy '1,1'. Similarly, if the sorting is to be done on the basis of first 3 fields, it will be: '-k 1,3'.

```
================================================================
awk

Print all country names in countries.txt that are longer than 10 characters
> cat countries.txt | awk 'length($0) > 10'

Print the sizes of each file in the current directory, and the total size:
> ls -l files | awk '{ x += $5 ; print $5 } END { print "total bytes: " x }'

What happen is you change print$5 to print$0?

================================================================
Let us now control the flow of execution in a multi-process environment:
Hint: use wait(). You can use "man 2 wait" to see the manual for wait() system call.

// Solution
#include <iostream>
#include <cstdio>
#include <unistd.h>
#include <sys/wait.h>

using namespace std;
int main ()
{
  pid_t pid;
  int  status;

  if(fork())
  {
    cout<<"I'm the Parent, and waiting\n"<<endl;
    pid = wait(&status);
    cout<<"I'm the Parent. my son's PID is "<<pid<<" my son's exit status is
"<<status<<endl;
  }
  else
  {
    cout<<"I'm the Son, and sleeping\n"<<endl;
    sleep(1);
    cout<<"I'm the Son, and exiting\n";
  }

  cout<<"Goodbye World\n";

  return(0);
}

================================================================
execl
execl replaces the calling process image with a new process image.

Create a file called a.cpp and write the following piece of code in it:

#include <iostream>
#include <cstdio>
#include <unistd.h>

using namespace std;

int main ()
{
        cout<<"Calling execl...\n\n";
        execl("/bin/cat", "cat", "./a.cpp", NULL);
        cout<<"Exiting!";
        return(0);
```

```
}
```

a. Add a new target to the Makefile to compile and link this code.
b. Run the executable.
- How many times “Exiting” is printed? Reason.

==========

Create a file called b.cpp and write the following piece of code in it:

```
#include <iostream>
#include <cstdio>
#include <unistd.h>
#include <sys/wait.h>

using namespace std;

int main()
{
        int status;

        if(fork())
        {
                cout<<"I'm Parent and don't do any thing"<<endl;
                wait(&status);
        }
        else
        {
                execl("/bin/ls","ls", "-l", NULL);
        }

   return 0;
}
```