```
/* CPSC 457 (Winter 2019)
 * Week 3 - Section 1
 * Multiprocess Programming
 * Sina Keshvadi
 *
 * Notes: No error handling!
 */
```

======================================

A process is a program in execution

1. Create a file called multiproc1.cpp and write the following piece of code:

```
#include <iostream>
#include <cstdio>
#include <unistd.h>
using namespace std;

int main()
{
   cout<<"Process ID id :"<<getpid()<<endl;
   cout<<"Parent Process id is "<<getppid<<endl;

   return 0;
}
```

a. Compile and run the code.

b. Run the executable multiple times on the same terminal.

- Does the process id (pid) change on each run? Reason
- Does the parent process id (pid) change on each run? Reason

======================================

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

fork() is the only way to create a process in Unix-like operating systems.

```
#include <iostream>
#include <cstdio>
#include <unistd.h>
using namespace std;

int main()
{
        pid_t pid;
        cout<<"Hello World"<<endl;

        pid = fork();

        if(pid > 0)
                cout<<"I'm the Parent and the PID of my child is "<<pid<<endl;
        else
                cout<<"I'm the Child"<<endl;

        cout<<"Goodbye World"<<endl;

   return 0;
}
```

a. Compile and run this code.

b. Run the executable.

- How many times “Goodbye World” is printed? Reason.

change the code to this:

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

using namespace std;

int main()
{
        pid_t pid;
        cout<<"Hello World"<<endl;
        pid=fork();
        pid=fork();
        pid=fork();
        cout<<"Goodbye World "<<pid<<endl;

   return 0;
}
```

- How many times “Hello World” is printed? Reason.
- How many times “Goodbye World” is printed? Reason.

======================================

Create a file called mp1.cpp and write the following piece of code:

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

using namespace std;

int main()
{
        fork();
        for(int i=1; i<=5; i++)
                cout<<i<<endl;

        return 0;
}
```

- review output
- change loop from 5 to 50 and then reviw the output

======================================

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

using namespace std;

int main()
{
        cout<<"Hello"<<endl;
        pid_t pid;
        pid = fork();

        if(pid<0)
                cout<<"Folk Failed";

        else if (pid==0)
        {
                cout<<endl<<"I am Child"<<endl;
                for(char i='a'; i<='g'; i++) // change loop to A to z
                        cout<<i<<endl;
                cout<<endl;
        }
        else
        {
                cout<<endl<<"I am Parent"<<endl;
                for(int i=1; i<=4; i++)
                        cout<<i<<endl;
                cout<<endl;
        }
        cout<<"Good Bye"<<endl;

   return 0;
}
======================================
```

This code show all directories.
Write this code in c.

```
#include <stdio.h>
#include <dirent.h>
#include <string.h>

void visitDir(const char *path, int depth) {
    DIR *directory = opendir(path);
    struct dirent* element = NULL;

    if(!directory) return;

    while(element = readdir(directory)) {
        for(int i=0; i<depth; i++)
            printf("\t");

        printf("%s\n", element->d_name);
        if(element->d_type == DT_DIR) {
            if(strcmp(element->d_name, ".") && strcmp(element->d_name, "..")){
                char buffer[1000];
                sprintf(buffer, "%s/%s", path, element->d_name);
                visitDir(buffer, depth+1);
            }
        }
    }
}

int main() {
    visitDir(".", 0);
    return 0;
}
======================================
```