### **13. g) STREAMS message/PIPEs/FIFO:pipe, popenand pcloseFunctions**

**Routines to let a parent and child synchronize. .**

**Objectives:**

1. To learn about STREAMS message/PIPEs/FIFO:pipe, popenand pcloseFunctions.

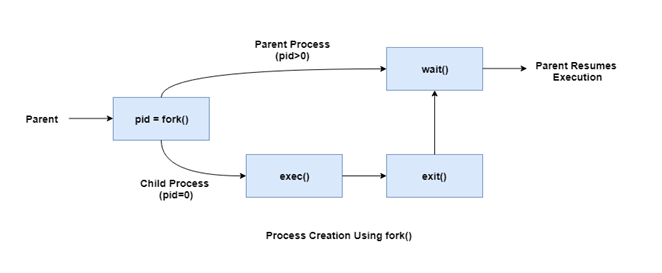
**Theory:**

Process synchronization in Linux involves providing a time slice for each process so that they get the required time for execution.

The process can be created using the fork() command in Linux. The creating process is called the parent process and the created process is the child process. A child process can have only one parent but a parent process may have many children. Both the parent and child processes have the same memory image, open files and environment strings. However, they have distinct address spaces.

If any process has more than one child processes, then after calling wait(), parent process has to be in wait state if no child terminates.   
If only one child process is terminated, then return a wait() returns process ID of the terminated child process.   
If more than one child processes are terminated than wait() reap any *arbitrarily child* and return a process ID of that child process.   
When wait() returns they also define exit status (which tells our, a process why terminated) via pointer, If status are not NULL.  
If any process has no child process then wait() returns immediately “-1”.

##### **Figure**



#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

static int pipefd1[2],pipefd2[2];

/\*Create two pipes\*/

void TELL\_WAIT(void){

if(pipe(pipefd1)<0 || pipe(pipefd2)<0){

perror("pipe");

exit(EXIT\_FAILURE);

}

}

void TELL\_PARENT(void){

/\* send parent a message through pipe, need you to complete\*/

printf("Child send message to parent!\n");

}

void WAIT\_PARENT(void){

/\* read message sent by parent from pipe, need you to complete\*/

printf("Child receive message from parent!\n");

}

void TELL\_CHILD(void){

/\* send child a message through pipe, need you to complete\*/

printf("Parent send message to child!\n");

}

void WAIT\_CHILD(void){

/\* read the message sent by child from pipe, need you to complete\*/

printf("Parent receive message from child!\n");

}

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

TELL\_WAIT();

pid\_t pid;

pid = fork();

//set a timer, process will end after 1 second.

alarm(10);

if(pid==0){

while(1){

sleep(rand()%2+1);

TELL\_CHILD();

WAIT\_CHILD();

}

}else{

while(1){

sleep(rand()%2+1);

WAIT\_PARENT();

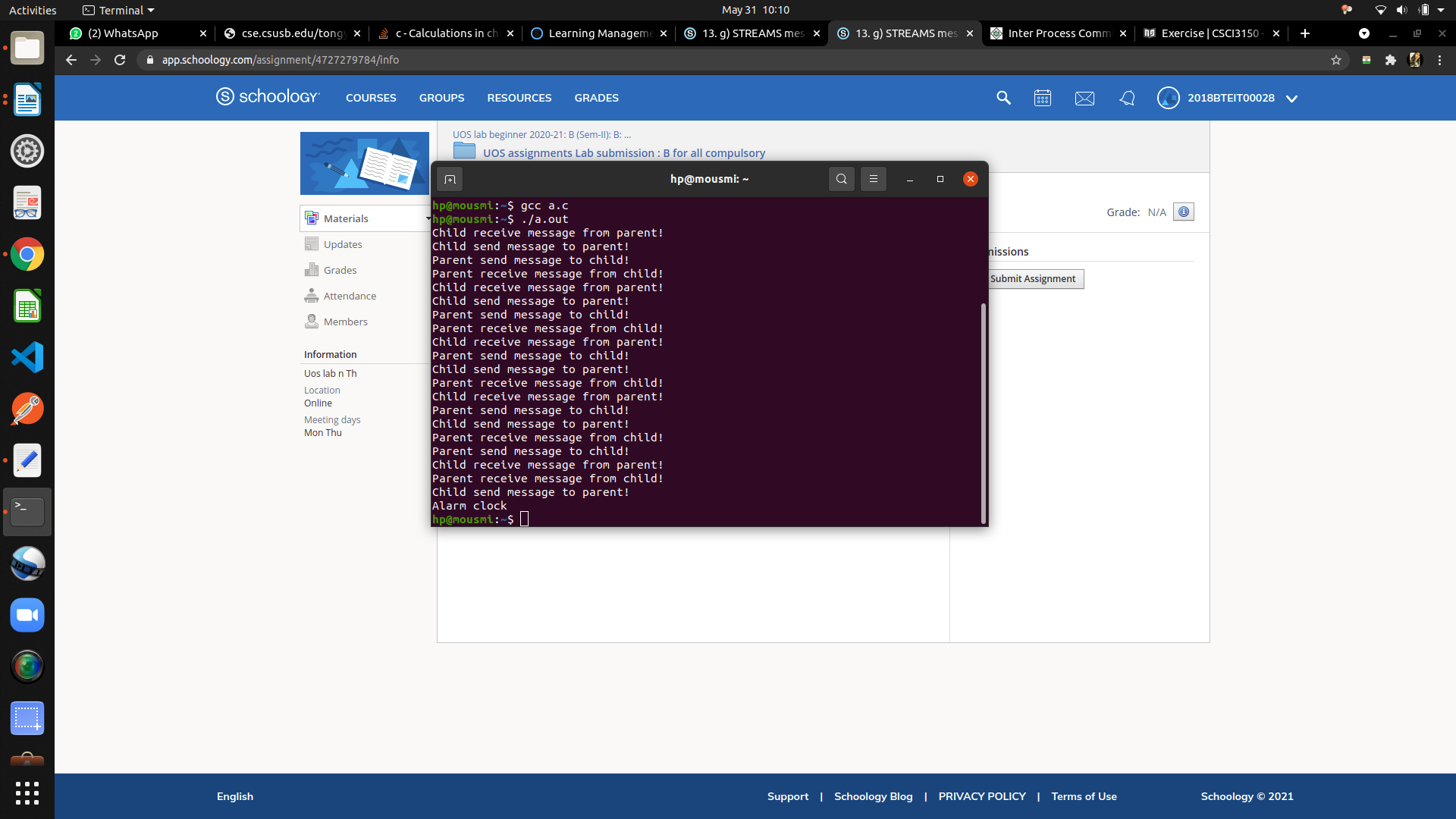
TELL\_PARENT();

}

}

return 0;

}



#### Conclusion:

Routines to let a parent and child synchronizeimplemented

#### References :

http://www.cse.cuhk.edu.hk/~ericlo/teaching/os/lab/6-IPC1/exercise-1.html