# Department of Computing

**CS 330: Operating Systems**

**BSCS 7C**

**Lab 4**: **Inter-process Communication using Pipes**

**CLO4(Develop programs to interact with OS components through its API)**

**Date: 18-10-2019**

**Time: 02:00 PM – 05:00 PM**

**Instructor: Dr. Fahad Javed**

**Lab 4**:**Inter-process Communication using Pipes**

**Introduction**

The purpose of this lab is to introduce you to intercrosses communication using pipes. Creating ``pipelines'' with the C programming language can be a bit more involved than our simple shell example. To create a simple pipe with C, we make use of the pipe () system call. It takes a single argument, which is an array of two integers, and if successful, the array will contain two new file descriptors to be used for the pipeline. After creating a pipe, the process typically spawns a new process (remember the child inherits open file descriptors).

**Objectives**

By the end of this lab you will be able to ensure bidirectional communication among processes

in consistent manners.

**Tools/Software Requirement**

* Linux OS installed on laptops or systems.
* GCC

**Description**

Creating a pipe in C language requires simple command which is a s follows.

SYSTEM CALL: pipe();

PROTOTYPE: int pipe( int fd[2] );

RETURNS: 0 on success

-1 on error: errno = EMFILE (no free descriptors)

EMFILE (system file table is full)

EFAULT (fd array is not valid)

NOTE: fd[0] is set up for reading, fd[1] is set up for writing

**Tasks**

Design a simple C program using ordinary pipes in which a parent and child processes exchange greeting messages. For example, the parent process may send the message “Hello Child Process”, and the child process may return “Hi Parent Process”. Use UNIX pipes to write this program. For reference, see http://tldp.org/LDP/lpg/node11.html. Note that this URL only shows an example of a unidirectional pipe. You need to have bidirectional behavior in your program for which you will need two pipes.

**Deliverables**

Submit the code.

**CODE:**

include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sched.h>

#include <sys/time.h>

#include <stdlib.h>

int main(void)

{

int fd[2],fd1[2], nbytes,nbytes2;

pid\_t childpid;

char string[] = "Hello World!\n";

char string2[] = "Hello, Broooo!\n";

char readbuffer[80];

char readbuffer2[80];

pipe(fd);

pipe(fd1);

if((childpid = fork()) == -1)

{

perror("fork");

exit(1);

}

if(childpid == 0)

{

/\* Child process closes up input side of pipe \*/

close(fd[0]);

/\* Send "string" through the output side of pipe \*/

write(fd[1], string, (strlen(string)+1));

close(fd1[1]);

nbytes2 = read(fd1[0], readbuffer2, sizeof(readbuffer2));

printf("Received string: %s", readbuffer2);

exit(0);

}

else

{

/\* Child process closes up input side of pipe \*/

close(fd1[0]);

/\* Send "string" through the output side of pipe \*/

write(fd1[1], string2, (strlen(string2)+1));

close(fd[1]);

nbytes = read(fd[0], readbuffer, sizeof(readbuffer));

printf("Received string: %s", readbuffer);

exit(0);

}

return(0);

}

**Output:**



# 