CIS 657 (POS) fall 2013 Lab 10 –

**I/0 System Calls**

# INTRODUCTION

In this lab, we are will implement client-server based basic calculator. The client process will be created by forking the server process. The client process will collect user input and pass it to server using pipe, similarly server will calculate the result and return to client for display using pipe.

**I/0 System Calls (Excerpts from “The C Programming Language” by Brian and Dennis)**

|  |
| --- |
| “In the UNIX operating system, all input and output is done by reading or writing ﬁle descriptors, because all peripheral devices, even the user’s terminal, are ﬁles in the ﬁle system. This means that a single, homogeneous interface handles all communication between a program and peripheral devices.” |

**1. Input and Output File Descriptor**

|  |
| --- |
| “Since input and output involving the user’s terminal are so common, special arrangements exist to make this convenient. When the command interpreter (the ‘‘shell’’) runs a program, it opens three ﬁles, with ﬁle descriptors 0, 1, and 2, called the standard input, the standard output, and the standard error output. All of these are normally connected to the terminal.” |

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Interface** | **Include Files** |
| open() | int open( char \*filename, int access, int permission ); | #include <fcntl.h> |
| read() | int read( int handle, void \*buffer, int nbyte ); | #include <fcntl.h> |
| write() | int write( int handle, void \*buffer, int nbyte ); | #include <fcntl.h> |
| close() | int close( int handle ); | #include <fcntl.h> |
| fcntl() | int fcntl(int fd, int cmd, ... /\* arg \*/ ); | #include <fcntl.h> |
| pipe() | int pipe(int filedes[2]); | #include <unistd.h> |
| select() | int select(int nfds, fd\_set \*readfds, fd\_set \*writefds, fd\_set  \*exceptfds, struct timeval \*timeout); | /\* According to POSIX.1-2001 \*/#include <sys/select.h>  /\* According to earlier standards \*/ #include <sys/time.h> #include <sys/types.h> #include <unistd.h> |
| dup() | int dup(int oldfd); int dup2(int oldfd, int newfd); | #include <unistd.h> |

**2. File Descriptor Operations**

The following file descriptor mangement and service methods are probably needed in this lab

For the select() operation which is used to monitor multiple file descriptors, several macros are defined in the system in order to mangae the set of descriptors. An example of the typical usage of select() with the corresponding macros to read data:

int ret;

fd\_set rd\_set;

int maxfd = descriptor\_1 > descriptor\_2 ? descriptor\_1:descriptor\_2; FD\_ZERO(&rd\_set);

FD\_SET(descriptor\_1,&rd\_set); FD\_SET(descriptor\_2,&rd\_set);

ret = select(maxfd + 1, &rd\_set, NULL, NULL, NULL);

//descriptor\_1 is ready if(FD\_ISSET(descriptor\_1,&rd\_set)){

//handle data from descriptor\_1

}

//descriptor\_2 is ready if(FD\_ISSET(descriptor\_2,&rd\_set)){

//handle data from descriptor\_2

}

# Relevant Links

# 1. [UNIX IO Programming](http://lcs3.syr.edu/faculty/bolazar/OS/uprog.pdf) 2. [Select call man page](http://www.unix.com/man-page/freebsd/2/select/)

# Tasks (90)

As mentioned, in this lab we will be implementing a client server based basic calculator. Below are the tasks for the lab.

Client to Server Pipe

Read User input

Write User input

All Calculations here

All UI Activities here

Server to Client Pipe

Read Result

Write Result

Server Process

Client Process

1. **Client Process (40)**

|  |  |
| --- | --- |
| Requirements | Points |
| Created by forking Server Process | 5 |
| Collect user input i.e. Two integer operands and one Character operator (+, -, \*, /, q for quit) | 5 |
| Send the collected input to server process via pipe (write()) | 15 |
| Get the response from server via pipe and display it (select() and read()) | 10 |
| Should exit properly on quit message | 5 |

1. **Server Process (40)**

|  |  |
| --- | --- |
| Requirements | Points |
| Extract the user input (integer operands and character operator) sent by client via pipe  (select() and read()) | 15 |
| Calculate the result and send to client via pipe (write()) | 15 |
| Error handling for invalid operator | 5 |
| Should exit properly on quit message | 5 |

1. **Other Requirements (10)**

|  |  |
| --- | --- |
| Requirements | Points |
| Program that compiles correctly | 5 |
| Program that runs without crash | 5 |

**Submission (10)**

Create and attach a README (txt/word/pdf) file at the end of the lab. It doesn't need to be comprehensive, but it should at least cover the following content:

* Which tasks are done, and which are not?
* What’s your basic idea to achieve these tasks?
* Where is your main function?
* Explain the files and the functions you created

If you can only finish some of the tasks in this project, please make sure that your code can at least be compiled and installed and also clearly state in the README file about the missing parts of your project.

**Checklist:** To submit your lab, you need to:

* Attach the ReadMe file
* Attach the C program you created
* Send this email to [cis657ta@nemmerle.syr.edu](mailto:cis657ta@nemmerle.syr.edu) with subject line “CIS657: Lab 10”