IPC Project Design Document

Ryan Abel

# Table of Contents

Introduction 1

Implementation 1

General Implementation Notes 1

Create Server Command 1

Abort Server Command 2

Display Status Command 2

Create Proc Command 2

Abort Proc Command 3

Conclusion 3

# 

# Introduction

The purpose of this assignment was to demonstrate exceptional knowledge of inter-process communication. This was to be achieved by designing and constructing a hierarchy of processes with the ability to communicate with one another. There were two options available to achieve communication. The first was using pipes and the second was using signals. I chose to use pipes in my implementation.

# Implementation

## General Implementation Notes

This program uses pipes to communicate between processes. The process manager handles input from the user. When a user enters a command the process manager interprets the command and acts accordingly.

## Create Server Command

The create server command has the following syntax:

createServer <MIN\_PROC> <MAX\_PROC> <NAME>

When createServer is called, the process manager creates a new pipe and then forks a new process. This process is responsible for spawning MIN\_PROC children. It is also responsible for setting it’s own name, it’s maximum processes (MAX\_PROC), it’s active processes, and maintaining a list of all of it’s children. Since this process must communicate with it’s children to kill them, createServer also creates a pipe for each of it’s children. Once the server has created it’s children, it waits for a command from the process manager. When the server receives a message it delegates to a function to execute the method.

## Abort Server Command

The abort server command has the following syntax:

abortServer <NAME>

When abortServer is called, the process manager opens a write pipe to all processes and puts the command in the pipe. Each server reads from its pipe. If the server name matches the name of the server to abort, the server will exit gracefully (along with all children).

## Display Status Command

The display status command has the following syntax:

displayStatus

The displayStatus command is responsible for displaying a hierarchy of the program’s state. This includes printing each server and each server’s children. The process manager sends a command to all servers. The server then prints its name along with its PID. Then it loops through it’s children and prints their PID.

## Create Proc Command

The create process command has the following syntax:

createProc <SRV\_NAME>

The create process command is responsible for creating a child process for the server specified in SRV\_NAME. The process manager sends the command to each server. If the server name matches, the server will then create a new pipe and fork a new process if it has not reached it’s maximum process number.

## Abort Proc Command

The abort process command has the following syntax:

abortProc <SRV\_NAME>

The abort process command is responsible for destroying a child process for the server specified in SRV\_NAME. The process manager sends the command to each server. If the server name matches, the server will then open a pipe with it’s child and write a “kill” command. The child will receive the command and exit gracefully. This only happens if it does not result in the server having less than it’s minimum allowed processes.

# Conclusion

This project was very helpful in understanding inter-process communication. It forced me to create several small experimental programs to investigate innards of system calls such as pipe() and fork().