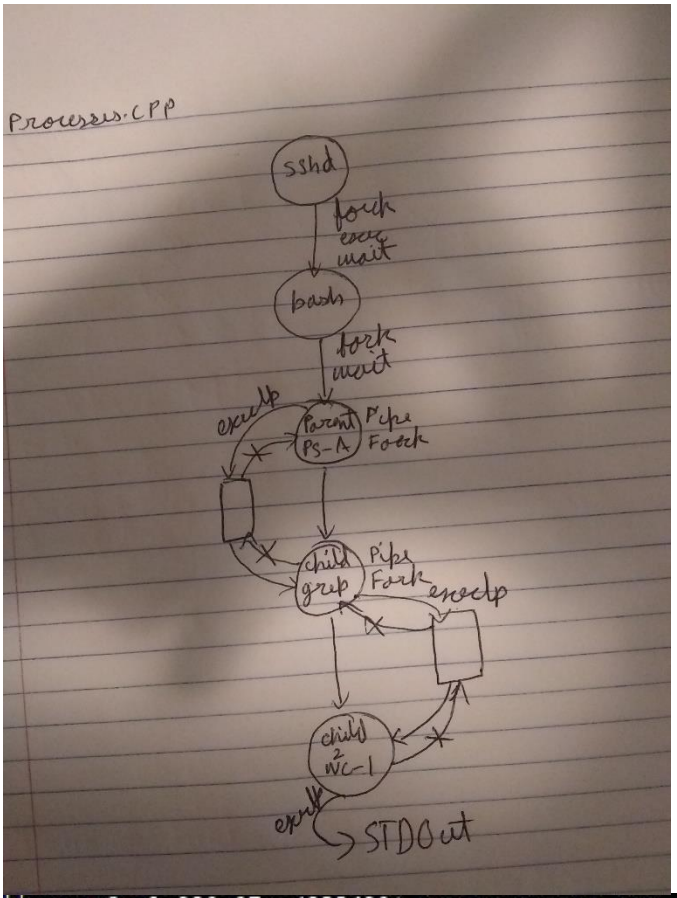
Khuzaima Mushtaq
CSS430 – Parsons

Program 1 Report

Part 1: processes.cpp

For processes.cpp I used two pipes and two forks.



As shown by the diagram I had two pipes between the parent and the child and then the child to the grandchild. A key thing to note is that I did not create both pipes at the start in parent but waited until child 1 to create the second pipe. I think it can be done either way, but I found this method to be better. The diagram also shows the openings and ends I had open/ close. In essence the parent calls exec ps -A then waits, the output of ps -A is redirected into the first pipe (fd1) and received by child 1 who has their end of the pipe open on stdin. Then child 1 writes the output of grep to child 2 who has their pipe open to listen on stdin, then child 2 calls exec wc -l on the input from grep and outputs it to std out which is displayed on the console. The parent waits until both the children are done, then exists.

Output:

```
khuzema@uw1-320-07:~/CSS430$ g++ processes.cpp -o processes
khuzema@uw1-320-07:~/CSS430$ ps -A | grep tty | wc -l
2
khuzema@uw1-320-07:~/CSS430$ ./processes tty
2
khuzema@uw1-320-07:~/CSS430$ ps -A | grep Sys | wc -l
0
khuzema@uw1-320-07:~/CSS430$ ./processes Sys
0
khuzema@uw1-320-07:~/CSS430$ ps -A | grep user | wc -l
0
khuzema@uw1-320-07:~/CSS430$ ./processes user
0
```

Part 2: Shell.java

How to Test:

Simply unzip the Khuzema_P1.zip file in the folder with all the ThreadOS components, use javac Shell.java command to compile my Shell.java. boot ThreadOS using java Boot. Use l Shell to execute Shell.java in ThreadOS

Once running type in valid .class programs in memory with any amount of args to execute them, use delimiters & and ; to separate commands

Algorithm:

I used a method String[] makeStrings(StringBuffer buf) { to parse the commands sent to my shell

makeStrings basically takes the user input and divides it into multiple strings based on delimiters.

For example a user input “PingPong a 50 ; PingPong b 10 & PingPong c 5”

Is seprerated into multiple strings and put into an array as such:

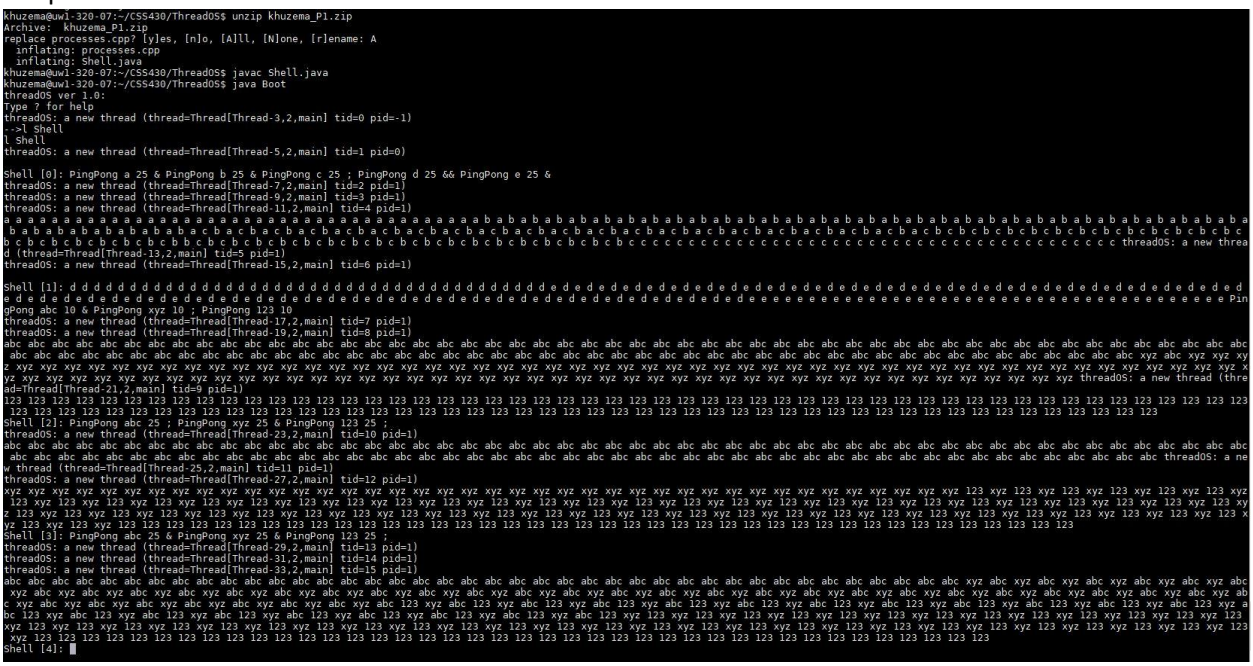
0	1	2	3	4	5
PingPong a 50	;	PingPong b 10	&	PingPong c 5	null

This is really helpful because then when I go to execute these commands because I can easily parse the command into args using the SysLib.stringToArgs commands included in SysLib and I also have access to the delimiter which is know is ALWAYS going to be in the next index of the array from the command. I execute the commands based on their delimiters until my array index hits null, then I stop, and my Shell asks for another command.

I did initially attempt to parse the array recursively for multiple & commands it was a fun experiment but it lead to the last command in a series of & commands to be executed first and then then recursively executing the other commands. So it would execute PingPong c 5 first then PingPong b 10 etc.

The biggest key to my programs is that I assume when I get my array from makeStrings() that it is formatted precisely as defined above other wise my whole run method would fall apart. If you encounter any bugs while running my code it is very likely that the cause would be in makeStrings().

Output:


khuzema@uw1-320-07:~/CSS430/ThreadOS$ unzip khuzema_P1.zip
Archive: khuzema_P1.zip
replace processes.cpp? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
inflating: processes.cpp
inflating: Shell.java
khuzema@uw1-320-07:~/CSS430/ThreadOS$ javac Shell.java
khuzema@uw1-320-07:~/CSS430/ThreadOS$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
l ->sl Shell
l Shell
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
Shell [0]: PingPong a 25 & PingPong b 25 & PingPong c 25 ; PingPong d 25 && PingPong e 25 &
threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=1)
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
threadOS: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=1)
threadOS: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=1)
threadOS: a new thread (thread=Thread[Thread-21,2,main] tid=9 pid=1)
Shell [2]: PingPong abc 25 ; PingPong xyz 25 & PingPong 123 25 ; .
threadOS: a new thread (thread=Thread[Thread-23,2,main] tid=10 pid=1)
threadOS: a new thread (thread=Thread[Thread-25,2,main] tid=11 pid=1)
threadOS: a new thread (thread=Thread[Thread-27,2,main] tid=12 pid=1)
Shell [3]: PingPong abc 25 & PingPong xyz 25 & PingPong 123 25 ;
threadOS: a new thread (thread=Thread[Thread-29,2,main] tid=13 pid=1)
threadOS: a new thread (thread=Thread[Thread-31,2,main] tid=14 pid=1)
threadOS: a new thread (thread=Thread[Thread-33,2,main] tid=15 pid=1)
Shell [4]: