COM S 352 Homework 2

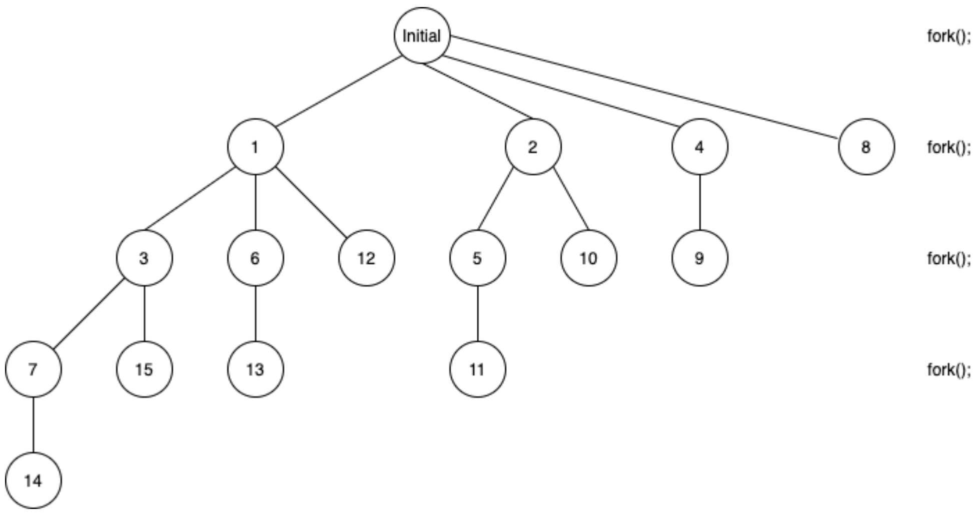
Alec Meyer

February 12, 2021

Question 1

First the operating system saves all data then transfers control the the kernal. The kernel saves the context of the old proccess in its PCB and then grabs the new process and loads the saved data into the new process. Finally the OS transfers control back to user mode.

Question 2



16 total processes created

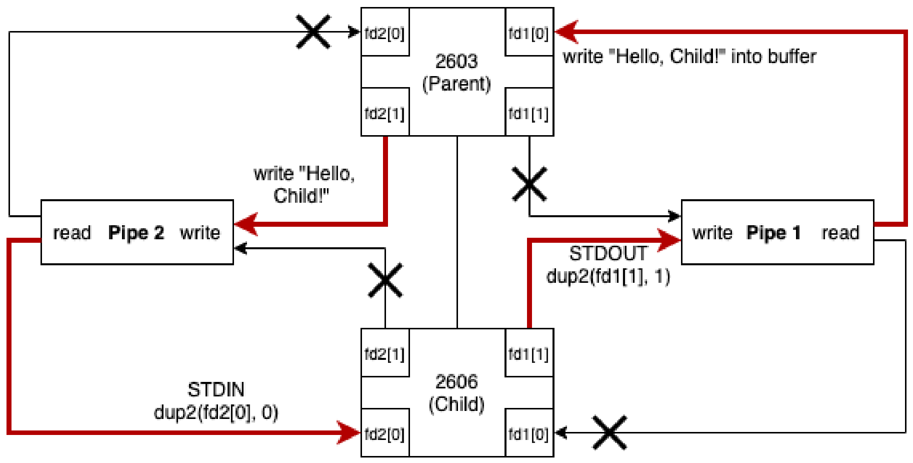
Question 3

The program executes, then runs into the *fork()* function, where it creates a child process. This child process will enter the *else if* statement because its pid is 0 then it will hit the *exec()* function. The *exec()* function replaces the childs memory space with a new program, in this case it is *ls*. Now that the child no longer exsists it will not reach the line, *printf("LINE J").* However, if the *exec()* function results in an error and memory is not replaced the child process will run the line *printf("LINE J").*

Question 4

A: 0	//child
B: 2606	//child
C: 2606	//parent
D: 2603	//parent

Question 5



The program will start by initializing its pipes and forks, creating a Parent (2603), Child (2606) and two pipes; fd1, fd2.
1. The Child calls *dup2(fd[1], 1)* which duplicates the file descriptor and redirects it to STDOUT
2. The Child calls *dup2(fd[0], 0)* which duplicates the file descriptor and redirects it to STDIN
3. The Child calls *execlp("cat", "cat", NULL)* which echos STDIN to STDOUT
4. The Child process terminates.
Described above creates a path from pipe 1 all the way to pipe 2. This path can be seen on the diagram represented by the red bold arrows. This path will allow the Parent to write to pipe 2 and read the message from pipe 1. Now that the parent has read the message it is stored into the buffer which prints "Hello, Child!".

Question 6

$\delta/(\delta + \sigma)$

System A:
$\delta = 20\text{ms}$
$\sigma = 1\text{ms}$

$20/(20 + 1)$
$= 95.24\%$

System B:
$\delta = 15\text{ms}$
$\sigma = 1\text{ms}$

$15/(15 + 1)$
=93.75%
The system with a slice of 20ms will be more efficient by 1.49%

Question 7

1. **Running to Waiting**
   During an I/O or event wait, The process is going to enter a waiting state because it is waiting for I/O input/output.
2. **Waiting to Ready**
   After waiting for an I/O response (completion) it will move to a ready state where it waits for reassignment.
3. **Running to Ready**
   The transition from running to ready will occur when an interrupt is triggered.