**Homework 1**

1. (10 points) Using the program shown below, explain what the output will be at LINE A.

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

int value = 10;

int main()

{

pid\_t pid;

  pid = fork();

  if (pid == 0) { /\* child process \*/

    value += 15;

    return 0;

  }

  else if (pid > 0) { /\* parent process \*/

    wait(NULL);

    printf("PARENT: value = %d",value); **/\* LINE A \*/**

    return 0;

  }

}

**LINE A OUTPUT: 10**

1. (10 points) Explain the circumstances under which the line of code marked printf(“LINE J”) in the program below will be reached.

  #include <sys/types.h>

  #include <stdio.h>

  #include <unistd.h>

  int main()

  {

   /\* fork a child process \*/

    pid\_t pid = fork();

    if (pid < 0) { /\* error occurred \*/

      fprintf(stderr, "Fork Failed");

      return 1;

    }

    else if (pid == 0) { /\* child process \*/

      execlp("/bin/ls","ls",NULL);

      printf("LINE J");

    }

    else { /\* parent process \*/

      /\* parent will wait for the child to complete \*/

      wait(NULL);

      printf("Child Complete");

    }

  return 0;

  }

This line will usually be skipped over, since it won’t be reached unless execlp were to fail.

1. (10 points) Including the initial parent process, how many processes are created by the program shown below:

#include <stdio.h>

#include <unistd.h>



int main()

{



   /\* fork a child process \*/



   fork();

   /\* fork another child process \*/



   fork();



   /\* and fork another \*/



   fork();

   return 0;

}



Total # of Processes: 8

1. (10 points) Using the program shown below, explain what the output will be at lines X and Y.

  #include <sys/types.h>

  #include <stdio.h>

  #include <unistd.h>

  #define SIZE 5

  int nums[SIZE] = {0,1,2,3,4};

  int main()

  {

   pid\_t pid = fork();

    if (pid == 0) {

     for (int i = 0; i < SIZE; i++) {

      nums[i] \*= -i;

      printf("CHILD: %d ",nums[i]); /\* LINE X \*/

     }

   }

    else if (pid > 0) {

     wait(NULL);

     for (i = 0; i lt; SIZE; i++)

     printf("PARENT: %d ",nums[i]); /\* LINE Y \*/

    }

    return 0;

  }

**LINE X OUTPUT:** 0 -1 -4 -9 -16

**LINE Y OUTPUT:** 0 1 2 3 4

1. Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Burst Time** | **Priority** | **Arrive Time** |
| *P1* | 2 | 2 | 3 |
| *P2* | 1 | 1 | 0 |
| *P3* | 8 | 4 | 2 |
| *P4* | 4 | 2 | 7 |
| *P5* | 5 | 3 | 5 |

a. (40 points) Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: **FCFS**, **SJF**, **non-preemptive** **Priority** (a larger priority number implies a higher priority), and **RR** (quantum = 2).



b. (5 points) What is the turnaround time of each process for each of the scheduling algorithms in part a.?

FCFS: t\_time = 8.8ms

SJF: t\_time = 8.6ms

Priority: t\_time = 8.6ms

RR: t\_time = 9.8ms

c. (5 points) What is the waiting time of each process for each of these scheduling algorithms?

FCFS: w\_time = 4.8ms

SJF: w\_time = 4.6ms

Priority: w\_time = 4.6ms

RR: w\_time = 5.8ms

d. (10 points) Which of the algorithms results in the minimum average waiting time (over all processes)?

Both SJF and non-preemptive Priority had the lowest average waiting time (4.6ms).