**CS 367 - Recitation #9 Exceptional Control Flows/Processes**

**Name Adilet Kuroda G#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Group Member Name:** Patrick Oppong, Jim Banya

**Today's Goals:** We want to get familiar with basic exceptional control flows and looking at processes.

**Work in groups of 2-3 students.** Every group will turn in what they've got to Blackboard.

**Grading is based on participation.** Get as much done as you can. You will also be given feedback in the form of a 'score' (1-3) and possibly some comments. This doesn’t affect your grade – it is solely for feedback. A score of 3 means everything looks great.   A score of two indicates some minor problems.  And a score of one indicates that there were some major issues. If you get a 1, don't panic - go see your prof or a GTA to get more extensive feedback.

**Processes and concurrency**

1. Assuming single-core machine, identify each pair of processes as concurrent or sequential processes. Put ‘C’ for concurrent and ‘S’ for sequential processes.

|  |  |  |
| --- | --- | --- |
| ../../../../../../Desktop/Pictur../../../../../../Desktop/Pictur../../../../../../Desktop/Pictur  Process **A B C D E** | A,B: | **S** |
| A,C: | **C** |
| A,D: | **S** |
| A,E: | **S** |
| B,C: | **S** |
| B,D: | **S** |
| B,E: | **C** |
| C,D: | **S** |
| C,E: | **C** |
| D,E: | **C** |

1. What is the mechanism that creates an illusion that each process has an exclusive use of the CPU?

**Concurrency**

1. How are processes created, loaded, and reaped? Explain using fork(), exec(), and wait().

**Fork creates the process. exec(): loads and runs the executable and wait() waits until child is done to ensure that child properly reaped.**

1. What is reaping and why is it needed?

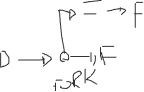
**Reaping is process where parent gets the exit status of child and terminates the child. It is important for synchronization. Reaping is used to ensure that there is no defunct (zombie) process.**

**Process creation and termination**

Consider the following C program.

|  |
| --- |
| fun1() {  printf("A\n");  if (fork()==0){  printf("B\n");  fork();  printf("C\n");  exit();  }  printf("D\n");  if (fork() == 0) printf("E\n");  printf("F\n"); } |

1. Draw the process graph of fun1(). Use the text area below or upload additional files for your graph.



Printf



1. In fun1(), how many times will each string ("A", "B", "C", "D", "E", "F") be output?  
   A: 1  
   B: 1  
   C: \_\_\_2\_\_\_  
   D: \_\_\_1\_\_\_  
   E: \_\_\_\_\_1\_  
   F: \_\_\_2\_\_\_



1. In fun1(), which of the following outputs are legal? (Disregard the missing \n in the output)
2. ADBCEFCF Legal



1. ABCDEFCF Legal
2. ADCEFBCF Illegal
3. ABCECDFF Illegal
4. ADEFBCFC Legal
5. Consider the following C program.   
     
    int main() {

if (fork() == 0) {

printf("a\n");

exit(0);

}

else {

printf("b\n");

waitpid(-1,null,0);

}

printf("C\n");

exit(0);

}

1. Draw the process graph. Use the text area below or upload additional files for your graph.



1. What are all of the possible output sequences of the program?

1. Redraw the process graph assuming the waitpid() call is removed. Use the text area below or upload additional files for your graph.

1. If the waitpid() call is removed, what are all of the possible outputs?