Homework 2

1. (10 points) Ex 3.8 Describe the actions taken by a kernel to context-switch between processes.

2. (10 points) Including the initial parent process, how many processes are created by the program shown below?

Draw a process tree to show the relationship between the processes.

```
int main()
{
      int i;
      for (i=0; i<2; i++) {
            fork();
            fork();
      }
      return 0;
}
```

3. (10 points) Ex 3.12 Explain the circumstances when the line of code marked printf("LINE J") in Figure 3.33 is reached.

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid;

   /* fork a child process */
   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;
}
```

Figure 3.33 When will LINE J be reached?

4. (10 points) Using the program in Figure 3.34, identify the values of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2603 and 2606, respectively.)

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid, pid1;

   /* fork a child process */
   pid = fork();

   if (pid < 0) { /* error occurred */
      fprintf(stderr, "Fork Failed");
      return 1;
   }
   else if (pid == 0) { /* child process */
      pid1 = getpid();
      printf("child: pid = %d",pid); /* A */
      printf("child: pid1 = %d",pid1); /* B */
   }
   else { /* parent process */
      pid1 = getpid();
      printf("parent: pid = %d",pid); /* C */
      printf("parent: pid1 = %d",pid1); /* D */
      wait(NULL);
   }

   return 0;
}
```

5. (10 points) The following program produces the output "Hello, Child!". Draw a diagram of all processes and pipes. The processes must be labeled with process ids 2603 and 2606 for the parent and child, respectively. The pipes must be labeled with their variable names from the code. The read and write ends of the pipes must be clearly labeled.

Explain the output of the program, use the diagram.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<string.h>
#include<sys/wait.h>

int main() {
    char buffer[100];
```

```
    int fd1[2];
    if (pipe(fd1) == -1) {
        return 1;
    }
    int fd2[2];
    if (pipe(fd2) == -1) {
        return 1;
    }

    pid_t pid = fork();
    if (pid < 0) {
        return 1;
    }

    if (pid == 0) {
        close(fd1[0]);
        close(fd2[1]);
        dup2(fd1[1], 1);
        dup2(fd2[0], 0);
        execlp("cat", "cat", NULL);
    } else {
        close(fd1[1]);
        close(fd2[0]);
        write(fd2[1], "Hello, Child!\n", 14);
        read(fd1[0], buffer, 100);
        close(fd1[0]);
        close(fd2[1]);
        printf("%s", buffer);
    }

    return 0;
}
```

6. (10 points) In a multitasking system, the context-switch time is 1ms and the time slice is 20ms. Will the CPU efficiency increase or decrease when we decrease the time slice to 15ms? Give both efficiencies and explain the difference.

7. (10 points) For each state transition, give an example event that will cause the transition:
(a) from running to waiting;
(b) from waiting to ready;
(c) from running to ready