Guía de ejercicios 2
Ejercicios sobre SO y procesos

1. ¿Cuántos procesos crea el siguiente código para MAX = 5? Considere que las funciones retornan con éxito.

```
main() {
  int pid, i, j;
  for (i = 0; i < MAX; i++)
    for (j = 0; j < MAX; j++)
      if ((pid = fork()) == 0) break;
}
```

2. Para el siguiente programa, dibuje el árbol de procesos indicando para cada nodo (proceso) lo que imprime. Asuma que no existen errores de ningún tipo.

```
main() {
  int i = 0;
  pid1 = fork();
  i = i + 1;
  pid2 = fork();
  i = i + 2;
  pid3 = fork();
  if (pid2 < 0 && pid1 != 0)
    printf("%d\n", i);
  else {
    printf(" %d\n", i+2);
  }
}
```

3. Una manera de medir el costo de un cambio de contexto es crear dos procesos que se comunican a través de dos pipes. El primer proceso ejecuta un write al primer pipe para luego hacer un read al segundo. Por otro lado, el segundo proceso ejecuta un read al primer pipe y un write al segundo. Este ciclo se repite, lo cual nos permite medir repetidas veces el cambio de contexto para luego obtener un promedio. ¿Qué problema puede distorsionar la medición en sistemas que soportan multiprogramación y multiprocesador?

4. El siguiente código es un extracto modificado de la última parte del scheduler de Linux, el que se ejecuta en el contexto de los procesos usuarios:

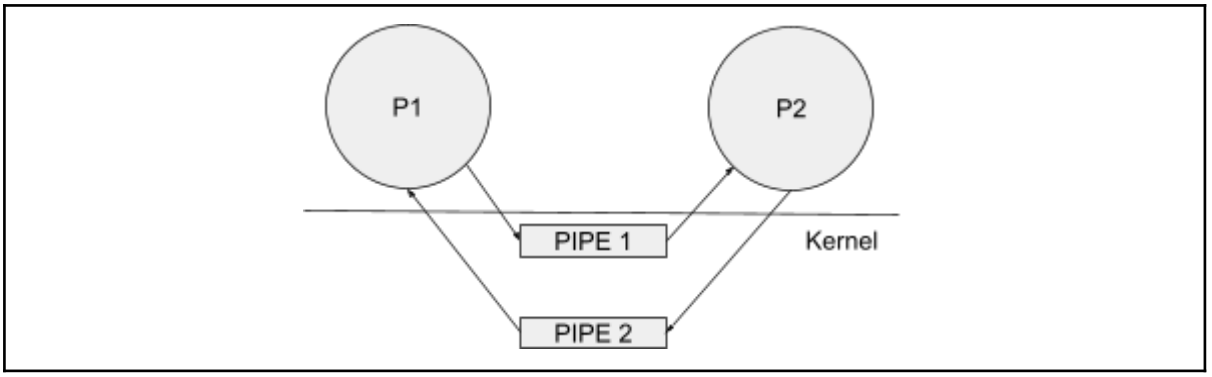
```
0 ...
1 if (prev != next) {
2   kernelstatistics.context_switch++;
3   switch_to(prev, next);
4 }
5 return;
```

prev y next son punteros a los PCBs de dos procesos. prev apunta al proceso que actualmente se está ejecutando en el procesador, y next al proceso que el scheduler ha seleccionado para que se ejecute a continuación. Si el proceso seleccionado es distinto al actual, entonces en la línea 2 se actualiza el contador del número de veces que se ha realizado cambio de contexto. Esto es sólo para registrar estadísticas. Luego, en la línea 3 se realiza el cambio de contexto. Explique qué sentido tiene la instrucción return (línea 5) justo después de haber realizado el cambio de contexto. Pregúntese quién y cuándo se ejecuta dicha instrucción.

5. Para cada proceso que se ejecuta en el siguiente programa, indique los valores que se imprimen por pantalla en las líneas 7 y 13. Asuma que no existen errores, que el PID del primer proceso es 10 y que los procesos hijos se enumeran secuencialmente a partir de 11 de acuerdo al orden de creación. Justifique su respuesta a través de un diagrama.

```
01 int main () {
02  int i, j;
03  pid_t pid;
04
05  i = j = 0;
06  pid = fork();
07  printf("pid=%d, i=%i", pid, i);
08  i = pid + 1;
09  if (pid == 0)
10    j = 1;
11  pid = fork();
12  j = pid - 1;
13  printf("pid=%d, i=%i", pid, i);
14 }
```

6. Utilizando fork() y pipe(), cree la siguiente estructura de comunicación entre procesos:
    a. Comunicación bidireccional (2 procesos, N procesos)



    b. Árbol binario (altura 3, altura N)