Laboratorio 2.

Integrantes:

- **Darla Albornoz**
- **Manfred Fernandez**
- **Rebeca Servellón Orellana**

1. Run process-run.py with the following flags: -l 5:100,5:100. What should the CPU utilization be (e.g., the percent of time the CPU is in use?) Why do you know this? Use the -c and -p flags to see if you were right.

```
rjso@ubuntu:~/Escritorio/Lab2$ ./process-run.py -l 5:100,5:100 -c
Time     PID: 0     PID: 1        CPU        IOs
  1     RUN:cpu      READY          1
  2     RUN:cpu      READY          1
  3     RUN:cpu      READY          1
  4     RUN:cpu      READY          1
  5     RUN:cpu      READY          1
  6        DONE     RUN:cpu         1
  7        DONE     RUN:cpu         1
  8        DONE     RUN:cpu         1
  9        DONE     RUN:cpu         1
 10        DONE     RUN:cpu         1
```

R/ En todas se usa el CPU el 100% de las veces.

2. Now run with these flags: ./process-run.py -l 4:100,1:0. These flags specify one process with 4 instructions (all to use the CPU), and one that simply issues an I/O and waits for it to be done. How long does it take to complete both processes? Use -c and -p to find out if you were right.

```
rjso@ubuntu:~/Escritorio/Lab2$ ./process-run.py -l 4:100,1:0 -c
Time     PID: 0     PID: 1        CPU        IOs
  1     RUN:cpu      READY          1
  2     RUN:cpu      READY          1
  3     RUN:cpu      READY          1
  4     RUN:cpu      READY          1
  5        DONE     RUN:io          1
  6        DONE    WAITING                     1
  7        DONE    WAITING                     1
  8        DONE    WAITING                     1
  9        DONE    WAITING                     1
 10*       DONE       DONE
```

R/ Le toma 10 ciclos para completar ambos procesos pues espera que termine el I/O.

3. Switch the order of the processes: -l 1:0,4:100. What happens now? Does switching the order matter? Why? (As always, use -c and -p to see if you were right)

```
rjso@ubuntu:~/Escritorio/Lab2$ ./process-run.py -l 1:0,4:100 -c
Time     PID: 0     PID: 1        CPU       IOs
  1      RUN:io      READY          1
  2     WAITING    RUN:cpu          1         1
  3     WAITING    RUN:cpu          1         1
  4     WAITING    RUN:cpu          1         1
  5     WAITING    RUN:cpu          1         1
  6*       DONE       DONE
```

R/ Especifica un proceso con una instrucción en CPU, y el otro emite 4 instrucciones de entrada y salida. El orden sí importa, ahora los procesos terminan en menos ciclos pues se realiza de manera paralela ambos procesos al tener primero la petición de I/O.

4. We'll now explore some of the other flags. One important flag is -S, which determines how the system reacts when a process issues an I/O. With the flag set to SWITCH ON END, **the system will NOT switch to another process while one is doing I/O, instead waiting until the process is completely finished**. What happens when you run the following two processes (-l 1:0,4:100 -c -S SWITCH_ON_END), one doing I/O and the other doing CPU work?

```
rjso@ubuntu:~/Escritorio/Lab2$ ./process-run.py -l 1:0,4:100 -c -S
SWITCH_ON_END
Time     PID: 0     PID: 1        CPU       IOs
  1      RUN:io      READY          1
  2     WAITING      READY                    1
  3     WAITING      READY                    1
  4     WAITING      READY                    1
  5     WAITING      READY                    1
  6*       DONE    RUN:cpu          1
  7        DONE    RUN:cpu          1
  8        DONE    RUN:cpu          1
  9        DONE    RUN:cpu          1
```

R/ El sistema espera a que los procesos de entrada y salida terminen completamente para seguir con el resto referentes al CPU, por lo que se gasta más tiempo durante la espera, como pasaba al tener primero la instrucción de CPU y luego la de I/O.

5. Now, run the same processes, but with the switching behavior set to switch to another process whenever one is WAITING for I/O (-l 1:0,4:100 -c -S SWITCH_ON_IO). What happens now? Use -c and -p to confirm that you are right.

```
rjso@ubuntu:~/Escritorio/Lab2$ ./process-run.py -l 1:0,4:100 -c -S
SWITCH_ON_IO
Time     PID: 0     PID: 1        CPU       IOs
  1      RUN:io      READY          1
  2     WAITING    RUN:cpu          1         1
  3     WAITING    RUN:cpu          1         1
  4     WAITING    RUN:cpu          1         1
  5     WAITING    RUN:cpu          1         1
  6*       DONE       DONE
```

R/ En caso de que el sistema esté esperando a que un proceso de I/O finalice, le indica que debe seguir con los otros procesos de CPU, entonces para cuando los procesos del CPU terminen, los de I/O también lo habrán hecho, como se realiza normalmente por default.

6. One other important behavior is what to do when an I/O completes. With -I IO RUN LATER, when an I/O completes, the process that issued it is not necessarily run right away; rather, whatever was running at the time keeps running. What happens when you run this combination of processes? (Run ./process-run.py -l 3:0,5:100,5:100,5:100 -S SWITCH_ON_IO -I IO_RUN_LATER -c -p) Are system resources being effectively utilized?

```
rjso@ubuntu:~/Escritorio/Lab2$ ./process-run.py -l 3:0,5:100,5:100,
5:100 -S SWITCH_ON_IO -I IO_RUN_LATER -c -p
Time     PID: 0     PID: 1     PID: 2     PID: 3        CPU
IOs
  1      RUN:io      READY      READY      READY           1

  2     WAITING    RUN:cpu      READY      READY           1
  1
  3     WAITING    RUN:cpu      READY      READY           1
  1
  4     WAITING    RUN:cpu      READY      READY           1
  1
  5     WAITING    RUN:cpu      READY      READY           1
  1
  6*      READY    RUN:cpu      READY      READY           1

  7       READY       DONE    RUN:cpu      READY           1

  8       READY       DONE    RUN:cpu      READY           1

  9       READY       DONE    RUN:cpu      READY           1

 10       READY       DONE    RUN:cpu      READY           1

 11       READY       DONE    RUN:cpu      READY           1

 12       READY       DONE       DONE    RUN:cpu           1
```

```
 16       READY       DONE       DONE    RUN:cpu           1

 17      RUN:io       DONE       DONE       DONE           1

 18     WAITING       DONE       DONE       DONE
  1
 19     WAITING       DONE       DONE       DONE
  1
 20     WAITING       DONE       DONE       DONE
  1
 21     WAITING       DONE       DONE       DONE
  1
 22*     RUN:io       DONE       DONE       DONE           1

 23     WAITING       DONE       DONE       DONE
  1
 24     WAITING       DONE       DONE       DONE
  1
 25     WAITING       DONE       DONE       DONE
  1
 26     WAITING       DONE       DONE       DONE
  1
 27*       DONE       DONE       DONE       DONE


Stats: Total Time 27
Stats: CPU Busy 18 (66.67%)
Stats: IO Busy  12 (44.44%)
```

R/ Los procesos que solicitan I/O se quedan esperando que los procesos que utilizan el CPU termine para ejecutarse incluso luego de estar listos para ejecutarse (por la instrucción run later), esto toma más ciclos para terminar la ejecución y gasta más recursos del CPU por la espera que se realiza.

7. Now run the same processes, but with -I IO_RUN_IMMEDIATE set, which immediately runs the process that issued the I/O. How does this behavior differ? Why might running a process that just completed an I/O again be a good idea?

```
rjso@ubuntu:~/Escritorio/Lab2$ ./process-run.py -l 3:0,5:100,5:100,
5:100 -S SWITCH_ON_IO -I IO_RUN_IMMEDIATE -c -p
Time      PID: 0     PID: 1     PID: 2     PID: 3        CPU
IOs
  1        RUN:io       READY      READY      READY          1

  2       WAITING    RUN:cpu      READY      READY          1
  1
  3       WAITING    RUN:cpu      READY      READY          1
  1
  4       WAITING    RUN:cpu      READY      READY          1
  1
  5       WAITING    RUN:cpu      READY      READY          1
  1
  6*       RUN:io       READY      READY      READY          1

  7       WAITING    RUN:cpu      READY      READY          1
  1
  8       WAITING       DONE    RUN:cpu      READY          1
  1
  9       WAITING       DONE    RUN:cpu      READY          1
  1
 10       WAITING       DONE    RUN:cpu      READY          1
  1
 11*       RUN:io       DONE      READY      READY          1

 12       WAITING       DONE    RUN:cpu      READY          1
  1
 13       WAITING       DONE    RUN:cpu      READY          1
  1
```

```
  8       WAITING       DONE    RUN:cpu      READY          1
  1
  9       WAITING       DONE    RUN:cpu      READY          1
  1
 10       WAITING       DONE    RUN:cpu      READY          1
  1
 11*       RUN:io       DONE      READY      READY          1

 12       WAITING       DONE    RUN:cpu      READY          1
  1
 13       WAITING       DONE    RUN:cpu      READY          1
  1
 14       WAITING       DONE       DONE    RUN:cpu          1
  1
 15       WAITING       DONE       DONE    RUN:cpu          1
  1
 16*         DONE       DONE       DONE    RUN:cpu          1

 17          DONE       DONE       DONE    RUN:cpu          1

 18          DONE       DONE       DONE    RUN:cpu          1


Stats: Total Time 18
Stats: CPU Busy 18 (100.00%)
Stats: IO Busy  12 (66.67%)
```

R/ De esta forma los procesos de I/O se realizan en paralelo a los que utilizan el CPU por lo que es buena idea para ahorrar ciclos.

8. Now runwith some randomly generated processes: -s 1 -l 3:50,3:50 or -s 2 -l 3:50,3:50 or -s 3 -l 3:50,3:50. See if you can predict how the trace will turn out. What happens when you use the flag -I IO_RUN_IMMEDIATE vs. -I IO_RUN_LATER? What happens when you use -S SWITCH_ON_IO vs. -S SWITCH_ON_END?

-s 1 -l 3:50,3:50 or **-s 2 -l 3:50,3:50** or **-s 3 -l 3:50,3:50**
-S SWITCH_ON_IO and **-I IO_RUN_IMMEDIATE**

```
darla@DESKTOP-TGH1R6R  ~/OS-2022  ./process-run.py -s 1 -l 3:50,3:50 -c -p
Time     PID: 0     PID: 1        CPU        IOs
  1     RUN:cpu      READY          1
  2      RUN:io      READY          1
  3     WAITING    RUN:cpu          1          1
  4     WAITING    RUN:cpu          1          1
  5     WAITING    RUN:cpu          1          1
  6     WAITING       DONE                     1
  7*     RUN:io       DONE          1
  8     WAITING       DONE                     1
  9     WAITING       DONE                     1
 10     WAITING       DONE                     1
 11     WAITING       DONE                     1
 12*       DONE       DONE

Stats: Total Time 12
Stats: CPU Busy 6 (50.00%)
Stats: IO Busy  8 (66.67%)
```

```
darla@DESKTOP-TGH1R6R  ~/OS-2022  ./process-run.py -s 2 -l 3:50,3:50 -c -p
Time     PID: 0     PID: 1        CPU        IOs
  1      RUN:io      READY          1
  2     WAITING    RUN:cpu          1          1
  3     WAITING     RUN:io          1          1
  4     WAITING    WAITING                     2
  5     WAITING    WAITING                     2
  6*     RUN:io    WAITING          1          1
  7     WAITING    WAITING                     2
  8*    WAITING     RUN:io          1          1
  9     WAITING    WAITING                     2
 10     WAITING    WAITING                     2
 11*    RUN:cpu    WAITING          1          1
 12        DONE    WAITING                     1
 13*       DONE       DONE

Stats: Total Time 13
Stats: CPU Busy 6 (46.15%)
Stats: IO Busy  11 (84.62%)
```

```
darla@DESKTOP-TGH1R6R  ~/OS-2022  ./process-run.py -s 3 -l 3:50,3:50 -c -p
Time     PID: 0     PID: 1        CPU        IOs
  1     RUN:cpu      READY          1
  2      RUN:io      READY          1
  3     WAITING     RUN:io          1          1
  4     WAITING    WAITING                     2
  5     WAITING    WAITING                     2
  6     WAITING    WAITING                     2
  7*    RUN:cpu    WAITING          1          1
  8*       DONE     RUN:io          1
  9        DONE    WAITING                     1
 10        DONE    WAITING                     1
 11        DONE    WAITING                     1
 12        DONE    WAITING                     1
 13*       DONE    RUN:cpu          1

Stats: Total Time 13
Stats: CPU Busy 6 (46.15%)
Stats: IO Busy  9 (69.23%)
```

-I IO_RUN_LATER and **-S SWITCH_ON_END**

```
darla@DESKTOP-TGH1R6R  ~/OS-2022  ./process-run.py -s 1 -l 3:50,3:50 -c -p -I IO_RUN_LATER -S SWITCH_ON_END
Time     PID: 0     PID: 1        CPU        IOs
  1     RUN:cpu      READY          1
  2      RUN:io      READY          1
  3     WAITING      READY                     1
  4     WAITING      READY                     1
  5     WAITING      READY                     1
  6     WAITING      READY                     1
  7*     RUN:io      READY          1
  8     WAITING      READY                     1
  9     WAITING      READY                     1
 10     WAITING      READY                     1
 11     WAITING      READY                     1
 12*       DONE    RUN:cpu          1
 13        DONE    RUN:cpu          1
 14        DONE    RUN:cpu          1

Stats: Total Time 14
Stats: CPU Busy 6 (42.86%)
Stats: IO Busy  8 (57.14%)
```

```
darla@DESKTOP-TGH1R6R  ~/OS-2022  ./process-run.py -s 2 -l 3:50,3:50 -c -p -I IO_RUN_LATER -S SWITCH_ON_END
Time     PID: 0     PID: 1        CPU        IOs
  1      RUN:io      READY          1
  2     WAITING      READY                     1
  3     WAITING      READY                     1
  4     WAITING      READY                     1
  5     WAITING      READY                     1
  6*     RUN:io      READY          1
  7     WAITING      READY                     1
  8     WAITING      READY                     1
  9     WAITING      READY                     1
 10     WAITING      READY                     1
 11*    RUN:cpu      READY          1
 12        DONE    RUN:cpu          1
 13        DONE     RUN:io          1
 14        DONE    WAITING                     1
 15        DONE    WAITING                     1
 16        DONE    WAITING                     1
 17        DONE    WAITING                     1
 18*       DONE     RUN:io          1
 19        DONE    WAITING                     1
 20        DONE    WAITING                     1
 21        DONE    WAITING                     1
 22        DONE    WAITING                     1
 23*       DONE       DONE

Stats: Total Time 23
Stats: CPU Busy 6 (26.09%)
Stats: IO Busy  16 (69.57%)
```

```
 darla@DESKTOP-TGH1R6R  ~/OS-2022  ./process-run.py -s 3 -l 3:50,3:50 -c -p -I IO_RUN_LATER -S SWITCH_ON_END
Time     PID: 0     PID: 1        CPU       IOs
  1     RUN:cpu      READY          1
  2      RUN:io      READY          1
  3     WAITING      READY                    1
  4     WAITING      READY                    1
  5     WAITING      READY                    1
  6     WAITING      READY                    1
  7*    RUN:cpu      READY          1
  8        DONE     RUN:io          1
  9        DONE    WAITING                    1
 10        DONE    WAITING                    1
 11        DONE    WAITING                    1
 12        DONE    WAITING                    1
 13*       DONE     RUN:io          1
 14        DONE    WAITING                    1
 15        DONE    WAITING                    1
 16        DONE    WAITING                    1
 17        DONE    WAITING                    1
 18*       DONE    RUN:cpu          1

Stats: Total Time 18
Stats: CPU Busy 6 (33.33%)
Stats: IO Busy  12 (66.67%)
```

R/ En los 3 casos se observa una disminución en el porcentaje de uso del CPU (CPU Busy) pues el I/O luego de estar listo para ejecutarse espera que el CPU termine para poder realizar la operación.