Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Sistemas Operativos 1

Nombre: Oward Francisco Alberí Sian Solis
Carné: 201901807

Hoja de Trabajo – CPU Scheduling

1. Explique cuál es la diferencia entre Scheduling Permisivo y No Permisivo.
    - Se diferencian en quién tiene el control sobre la asignación de tiempo del procesador: en el "scheduling" permisivo, los procesos pueden decidir ceder el control voluntariamente, mientras que en él no permisivo, el sistema operativo puede forzar a los procesos a ceder el control según su programación interna.

2. ¿Cuál de los siguientes algoritmos de Scheduling podría provocar un bloqueo indefinido? Explique su respuesta.
    a. First-come, first-served
        a. Ejecuta los procesos en el orden en el que llega a la cola de listos. Si un proceso que llega primero es largo y ocupa el CPU durante mucho tiempo, los procesos que llegan después deben esperar en la cola de listos hasta que ese proceso termine..
    b. Shortest job first
    c. Round robin
    d. Priority

3. De estos dos tipos de programas:
    a. I/O-bound (un programa que tiene más I/Os que uso de CPU)

b. CPU-bound (un programa que tiene más uso de CPU que I/Os)

¿Cuál tiene más probabilidades de tener cambios de contexto voluntarios y cuál tiene más probabilidades de tener cambios de contexto no voluntarios? Explica tu respuesta.

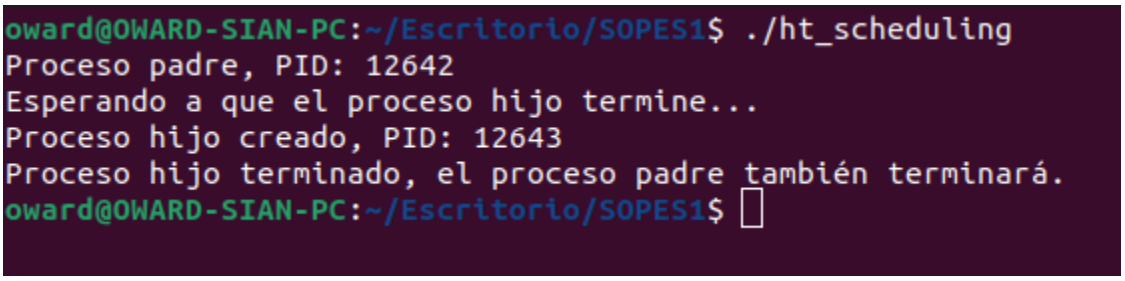
c. I/O-bound (un programa que tiene más I/Os que uso de CPU)

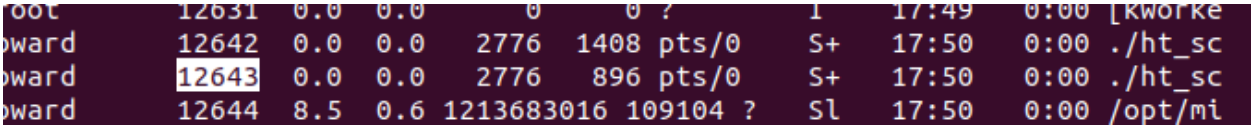
   a. Los IO-bound tienen más posibilidades de de tener cambios de contexto voluntario porqué ingresan muchas operaciones de salita y entrada mientras el programa realiza esto puede ceder el control al procesador de menra voluntaria mientras espera que se termine la entrada y salida de datos.

d. CPU-bound (un programa que tiene más uso de CPU que I/Os)

   a. Los CPU -bound utilizan de una manera más intensa el cpu y por tiempos largos, el sistema operativo puede intervenir para dar a otros procesos la oportunidad de ejecutarse.

4. Utilizando un sistema Linux, escriba un programa en C que cree un proceso hijo (fork) que finalmente se convierta en un proceso zombie. Este proceso zombie debe permanecer en el sistema durante al menos 10 segundos.
   Los estados del proceso se pueden obtener del comando: ps -l





GNU nano 6.2 ht_scheduling.c

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

int main() {
    pid_t pid;

    // Crear un proceso hijo
    pid = fork();

    if (pid < 0) {
        // Error al crear el proceso hijo
        perror("Error al crear el proceso hijo");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        // Código ejecutado por el proceso hijo

        printf("Proceso hijo creado, PID: %d\n", getpid());

        // El proceso hijo se duerme durante 10 segundos
        sleep(10);

        // Salir del proceso hijo
        exit(EXIT_SUCCESS);
    } else {
        // Código ejecutado por el proceso padre

        printf("Proceso padre, PID: %d\n", getpid());

        // Esperar a que el proceso hijo termine (se convierta en zombie)
        printf("Esperando a que el proceso hijo termine...\n");
        wait(NULL);
        printf("Proceso hijo terminado, el proceso padre también terminará.\n");
    }

    return 0;
```

^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^J Justificar ^/ Ir a línea