Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Sistemas Operativos 1
Pablo Josué Barahona Luncey
202109715

Hoja de Trabajo – CPU Scheduling

1. **Explique cuál es la diferencia entre Scheduling Permisivo y No Permisivo.**

   Scheduling Permisivo:

   - Permite que los procesos se ejecuten incluso si no tienen todos los recursos que necesitan.
   - Puede provocar la degradación del rendimiento del sistema, ya que los procesos que esperan recursos pueden bloquear a otros.
   - Se utiliza generalmente en sistemas donde la respuesta rápida es más importante que la eficiencia.

   Scheduling No Permisivo:

   - No permite que un proceso se ejecute a menos que tenga todos los recursos que necesita.
   - Evita la degradación del rendimiento del sistema, ya que los procesos no se bloquean entre sí.
   - Se utiliza generalmente en sistemas donde la eficiencia es más importante que la respuesta rápida.

2. **¿Cuál de los siguientes algoritmos de Scheduling podría provocar un bloqueo indefinido? Explique su respuesta.**
    a. First-come, first-served
    b. Shortest job first
    c. Round robin
    d. Priority

El algoritmo de prioridad puede provocar un bloqueo indefinido si un proceso de alta prioridad se queda sin recursos y no puede completarse. Esto puede impedir que los procesos de baja prioridad se ejecuten, ya que no pueden obtener los recursos que necesitan.

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.

    Los programas I/O-bound son más propensos a tener cambios de contexto voluntarios porque ceden la CPU voluntariamente cuando esperan a que se complete una operación de E/S. Los programas CPU-bound son más propensos a tener cambios de contexto no voluntarios porque no tienen tiempo para ceder la CPU a otros procesos.

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

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main() {
 int pid = fork();
 if (pid == 0) {
  printf("Proceso hijo creado con PID: %d\n", getpid());
  sleep(10);
  exit(0);
 } else if (pid > 0) {
  // Proceso padre
  printf("Proceso padre con PID: %d\n", getpid());
 } else {
  perror("Error al crear el proceso hijo");
  return 1;
 }
 return 0;
}
```