Guía de Procesos en Sistemas Operativos (Computación II - UM)

1. Fundamentos de Procesos

Un proceso es una instancia de un programa en ejecución. Tiene atributos como PID (identificador único), estado (ej: listo, ejecutando, esperando), prioridad, y contexto.

Diferencias entre proceso y programa:

- Programa: conjunto de instrucciones (estático)

- Proceso: ejecución del programa (dinámico)

En sistemas modernos, cada proceso tiene su propio espacio de direcciones en memoria, aislado de otros procesos.

2. Modelo de Procesos en UNIX/Linux

En UNIX/Linux los procesos forman una jerarquía: los procesos crean otros procesos (hijos) usando fork().

El proceso inicial del sistema es `init` o `systemd` (PID = 1), el cual adopta procesos huérfanos.

Herramientas útiles: `ps`, `htop`, `pstree`.

3. Manipulación de Procesos con Python

Python permite manipular procesos usando el módulo `os`:

- `os.fork()` crea un nuevo proceso (copia del actual)

- `os.exec*()` reemplaza el proceso actual por uno nuevo

- `os.wait()` espera que un proceso hijo termine

4. Procesos Zombis y Huérfanos

Un proceso zombi es uno que terminó, pero su padre no leyó su estado con `wait()`. Permanece en la tabla de procesos.

Un proceso huérfano es un hijo cuyo padre terminó antes. Es adoptado por `systemd`.

5. Ejercicios Prácticos en Python

- Crear un hijo que imprima su PID

- Crear múltiples hijos y sincronizar

- Usar `exec()` para reemplazar procesos

- Simular un proceso zombi y observarlo con `ps` o `htop`