AYUDANTÍA T2

Germán Leandro Contreras Sagredo
Ricardo Esteban Schilling Broussaingaray

IIC2333 [2018-2] - Sistemas Operativos y Redes

INTRODUCCIÓN

INTRODUCCIÓN

Los objetivos de esta ayudantía son:

1. Repasar la utilidad de `fork`, `exec` y `wait`.
2. Aclarar el funcionamiento esperado del **juego de la vida**.
3. Resolver las dudas que surjan durante la explicación de lo pedido.

FORK, EXEC, WAIT

FORK

- Se usa para crear nuevos procesos que se ejecutan concurrentemente con el proceso padre o creador.
- Se ejecuta el mismo código desde la línea donde se hizo `fork`.
- Es un proceso distinto.

```
void main() {
    int pid = fork();
    if (pid == 0) {
        printf("Nací!\n");
    }
    else {
        printf("Nuevo hijo %d!\n", pid);
    };
};
```

¿Cómo sé si un proceso es padre o hijo?

Volvamos a ver el código anterior:

```
void main() {
    int pid = fork();
    if (pid == 0) {
        printf("Nací!\n");
    }
    else {
        printf("Nuevo hijo %d!\n", pid);
    };
};
```

En `pid` el padre recibe el `pid` del hijo, mientras que el hijo recibe un 0.

EXEC

- Se usa para reemplazar todo el contenido de un proceso por otro programa.
- Sirve para ejecutar distintas cosas después de haber hecho `fork`.

```
void main() {
    if (fork() == 0) {
        printf("Nací!\n");
        char cmd[10] = "ls";
        char *args[] = {cmd, NULL};
        execvp(cmd, args);
    };
};
```

El ejemplo anterior corre el programa `ls`.

WAIT

- Se usa para esperar la finalización de un proceso hijo.
- Se podría usar para reasignar recursos escasos.

```
if ((pid = fork()) == 0) {
    sleep(2);
    exit(1);
} else do {
    if ((pid = waitpid(pid, &status, 1)) == 0) {
// Es 0 si el hijo especificado aún no hace exit
        printf("Still running!\n");
        sleep(1);
    } else {
        printf("Exit!\n");
    }
} while (pid == 0);
```

JUEGO DE LA VIDA

JUEGO DE LA VIDA

- Los tableros pueden ser de largo arbitrario (`int`).
- El programa **no debe caerse** cuando uno hace CTRL+C.
- Para enviar una señal a otro proceso, pueden usar la `syscall`:

```
kill(pid_t pid, int sig)
```

- El archivo `.csv` que deben crear puede tener nombre arbitrario.
- El nombre de cada tablero no será mayor a 256 caracteres.
- Arbitrariamente, el tiempo de simulación comienza en `t = 0`.

JUEGO DE LA VIDA

- Tener cuidado con detección de loops:
  1. Loops de periodo mayor a 4 **no** deben ser detectados.
  2. El tablero termina en el momento que se detecta un loop.
- Tener cuidado con los tiempos de término:
  1. Cuando se detecta una condición de término, el tablero termina inmediatamente.
- El orden de impresión en consola debe ser igual al orden en el que los procesos van terminando.
- El orden del archivo `.csv` **debe** ser igual al orden del input.

FIN