Universidad Nacional de San Agustín.
Escuela Profesional de Ingeniería de Sistemas

EJERCICIO DE LABORATORIO
SISTEMAS OPERATIVOS

Docente: Pablo Calcina Ccori
2 de octubre de 2020

1. COMPETENCIA DEL CURSO

La comprensión intelectual y la capacidad de aplicar las bases matemáticas y la teoría de la informática

2. COMPETENCIA DE LA PRÁCTICA

Identifica y aplica los principios de creación de procesos (fork, exec y waitpid). También se aplican los conocimientos de tratamiento de señales en el sistema Linux.

3. MATERIALES

- Computadora con el sistema operativo Linux.

4. EJERCICIOS PROPUESTOS

Crear los siguientes programas:

4.1. padre.c

- Será responsable de crear un proceso hijo usando la llamada *fork()*.
- Después de la creación de cada proceso hijo, por medio de la llamada *exec* (*execlp*) reemplazará el código del programa hijo, para ejecutar el programa *hijo.c*
- El programa deberá esperar a que termine la ejecución de los programas *hijo.c* y *nieto.c*. Usar *waitpid()*.

4.2. hijo.c

- El programa *hijo.c* creará un proceso hijo, por medio de la llamada *fork()*.
- Después de la creación de cada proceso hijo, por medio de la llamada *exec* (*execlp*) reemplazará el código del programa hijo, para ejecutar el programa *nieto.c*

- El programa hijo enviará al programa nieto (del cual conoce el PID), cada una de las señales listadas en ***Señales soportadas.***
- Detectar si el proceso *nieto* muere (Hint: detectar la señal respectiva), y lanzarlo de nuevo, hasta completar el envío de todas las señales requeridas.

4.3. nieto.c

- En este programa serán redefinidos los manejadores (*handlers*) de las señales indicadas en la sección ***Señales soportadas.***
- El tratamiento de señales debe usar la función *sigaction()*. No usar la función *signal()*.
- Para cada señal recibida mostrar la siguiente información, obtenida de *siginfo_t* (cuando esté disponible)
  - `Signal number (si_signo)`
  - `Signal code (si_code)`
  - `errno value (si_errno)`
  - `Sending process ID (si_pid)`
- Después de mostrar la información de la señal recibida, cuando sea posible, el programa continuará la ejecución. Por ejemplo, después de un SIGSEGV, el programa no puede recuperarse (Un ejercicio avanzado interesante sería pensar en qué hacer para recuperarse después de una señal SIGSEGV).

Señales soportadas

- `SIGHUP  (1)`
- `SIGINT  (2)`
- `SIGQUIT (3)`
- `SIGILL  (4)`
- `SIGABRT (6)`
- `SIGFPE  (8)`
- `SIGKILL (9)`
- `SIGSEGV (11)`
- `SIGPIPE (13)`
- `SIGALRM (14)`
- `SIGTERM (15)`
- `SIGCHLD (17)`
- `SIGSTOP (19)`
- `SIGTSTP (20)`

5. ENTREGABLES

5.1. **Arquivo comprimido conteniendo los archivos padre.c, hijo.c y nieto.c. También deberán incluirse un archivo Makefile, que compila todos los programas y ejecuta el programa padre.c.**

5.2. **El código deberá contener comentarios explicando su funcionamiento, en particular al invocar llamadas del sistema (fork, execlp, waitpid, sigaction) y el uso de siginfo_t.**

5.3. **En un archivo README.md colocar los nombres de los participantes si el trabajo es hecho en pares. En este archivo también colocar una explicación sucinta del programa. Mencionar, si hubiera, comportamientos inesperados en el tratamiento de señales en su sistema.**

5.4. **El nombre del archivo comprimido tendrá el siguiente formato:** `Lab-SO-Nombre-Apellidos.(tar.gz|zip|tar.bz2)`

Obs. No hay ningún impedimento para incluir archivos adicionales (.h o .c), desde que sea justificado. Por ejemplo, para crear funciones de uso común entre los diferentes archivos.

6. RÚBRICA

Criterios	Muy bueno	Bueno	Regular	Malo
Código	Realiza todas las tareas solicitadas sin errores **16 pts**	Realiza la mayoría de las tareas solicitadas o todas pero con errores **10 pts**	Realiza pocas tareas solicitadas y/o el programa tiene errores. **6 pts**	No entrega el trabajo **0 pts**
Explicación del código	Describe de forma clara y concisa el código implementado, usando nombres claros y explicativos para las variables y formatación que ayuda a la legibilidad. **4 pts**	Describe sin claridad el código implementado o no usa nombres claros y explicativos o la formatación no ayuda a la legibilidad. **2 pts**	Describe sin claridad el código implementado y/o no usa nombres claros y explicativos y/o la formatación no ayuda a la legibilidad. **1 pto**	Sin explicación del código. Usa nombres de variables y formatación que dificultan el entendimiento. **0 pts**