# Teoría de Sistemas Operativos

# Tarea #3

# Procesos y Señales en Linux

Escuela de Ingeniería Civil Informática

Universidad de Valparaíso

## Objetivo

Ambientar al alumno en el API de procesos y señales en Linux.

## Instrucciones

Su trabajo deberá ser entregado en el servidor en su cuenta en GitHub.com. El nombre del repositorio debe ser SSOO-tarea03

+ /SSOO-tarea03

- README.md

- Makefile

- src/

El archivo **README.md** es un archivo en markdown, donde se debe explicar el diseño de su solución, funciones utilizadas, etc. Además, debe especificar el nombre del autor y su correo institucional. En el directorio **src/** están los códigos fuentes de su proyecto.

## Trabajo a realizar

Esta tarea tiene como objetivo introducirlo al manejo de procesos y uso de señales[[1]](#footnote-1) en C++. Escriba un programa en C++17 que muestre por pantalla, la secuencia de Fibonacci y la serie de números impares ∑(2n-1). Para esto genere 2 procesos en paralelo que realicen la operación solicitada, de la siguiente manera:

1. Un proceso padre deberá de mostrar la secuencia de Fibonacci y un proceso hijo deberá de mostrar la Serie de números impares. En el ejemplo que se ve a continuación el proceso padre 666 muestra la secuencia de Fibonacci {1,2,3,5,…} y el proceso hijo 667 muestra la serie de números impares {1,3,5,7,…}.

host# ./main

Valor Fibonacci: 1 – PPIDE = 666

Valor Impar: 1 – PID = 667

Valor Fibonacci: 2 – PPIDE = 666

Valor Impar: 3 – PID = 667

Valor Fibonacci: 3 – PPIDE = 666

Valor Impar: 5 – PID = 667

Valor Fibonacci: 5 – PPIDE = 666

Valor Impar: 7 – PID = 667

1. El Código no puede parar su ejecución por ningún tipo de señal como se muestra en el siguiente recuadro (ninguna acción por consola debe detener el código: Ej: ctrl+C, ctrt+Z, ctrt+\). La única forma de que el código termine su ejecución es al completar 50 veces la iteración del ciclo o enviando una señal desde otra consola como se explica en el siguiente punto (3).

host# ./main

Valor Fibonacci: 1 – PPIDE = 666

Valor Impar: 1 – PID = 667

^C

Detencion no permitida

Valor Fibonacci: 2 – PPIDE = 666

Valor Impar: 2 – PID = 667

^Z

Detencion no permitida

Valor Fibonacci: 3 – PPIDE = 666

Valor Impar: 5 – PID = 667

**Ayuda:** Considera utilizar la sentencia this\_thread::sleep\_for(std::chrono::seconds(1))para controlar el tiempo de salida de los números.

1. Para detener la ejecución deberá de abrir otra terminal y enviar la señal kill -s SIGUSR2 667 para poder detener al hijo o kill -s SIGUSR1 666 para detener al padre. Si uno de los procesos es detenido el otro deberá de continuar con su ejecución.

host# kill -s SIGUSR1 666

host# ./main

Fibonacci: 1 – PPIDE = 666

Impar: 1 – PID = 667

Fibonacci: 2 – PPIDE = 666

Impar: 3 – PID = 667

Fibonacci: 3 – PPIDE = 666

Impar: 5 – PID = 667

Fibonacci: 5 – PPIDE = 666

Impar: 7 – PID = 667

Impar: 9 – PID = 667

Impar: 11 – PID = 667

Impar: 13 – PID = 667

host# kill -s SIGUSR2 667

host# ./main

Fibonacci: 1 – PPIDE = 666

Impar: 1 – PID = 667

Fibonacci: 2 – PPIDE = 666

Impar: 3 – PID = 667

Fibonacci: 3 – PPIDE = 666

Impar: 5 – PID = 667

Fibonacci: 5 – PPIDE = 666

Impar: 7 – PID = 667

Fibonacci: 8 – PPIDE = 666

Fibonacci: 13 – PPIDE = 666

Fibonacci: 21 – PPIDE = 666

## Condiciones de revisión

Se clonará el repositorio de su trabajo. Para esto, habrá un formulario disponible en el aula virtual donde deberá rellenar con su nombre y URL de su repositorio Git. Una vez clonado, se verificará la estructura de su proyecto se ajuste a las instrucciones entregadas. Para compilar el código debe de crear un archivo Makefile para crear el archivo ejecutable.

host# git clone https://github.com/usuarioGIT/SSOO-tarea03.git dirUser

host# cd dirUser

Una vez comprobado esto, se ejecutará, en una consola lo siguiente:

1. Al ejecutar su código la salida es la correcta **(3 pts)**

host# make

host# ./main

Valor Fibonacci: 1 – PPIDE = 666

Valor Impar: 1 – PID = 667

Valor Fibonacci: 2 – PPIDE = 666

Valor Impar: 3 – PID = 667

Valor Fibonacci: 3 – PPIDE = 666

Valor Impar: 5 – PID = 667

1. Posteriormente se tratará de detener el código con las instrucciones ctrl+C, ctrt+Z y ctrt+\ y la ejecución del código no debe de ser interrumpida **(3tps)**

host# ./main

Valor Fibonacci: 1 – PPIDE = 666

Valor Impar: 1 – PID = 667

^C

Detencion no permitida

Valor Fibonacci: 2 – PPIDE = 666

Valor Impar: 2 – PID = 667

^Z

Detencion no permitida

Valor Fibonacci: 3 – PPIDE = 666

^\

Detencion no permitida

Valor Impar: 5 – PID = 667

1. Finalmente se ejecutarán los comandos kill -s SIGUSR1 666 y kill -s SIGUSR2 667 en una consola aparte para detener al proceso padre o al proceso hijo, recuerde que, si alguno de los procesos es detenido, el otro debe de continuar **(4pts).**

**Observación**: En el caso que la ejecución del programa no es el previsto o no es adecuado, se calificará con nota 1,0.

1. ProcesosySenales.pdf – Gabriel Astudillo [↑](#footnote-ref-1)