# INFO198 - SISTEMAS OPERATIVOS
# Prueba 1 - parte práctica

2023-02 (Dr. Luis Veas C.)

La prueba 1 consta de 2 partes, una teórica y una práctica.

- La parte teórica se realizará el día jueves 28/09/2023 en horario de clases
- La parte práctica se entrega a los alumnos el día viernes 08/09/2023, y se debe entregar al profesor a más tardar el día miércoles 27/09/2023 a las 23:59 horas. Esta sección podrá realizarse de dos personas (no es obligación). La forma de entrega es vía email, bajo el siguiente formato:

    asunto: INFO198 - SISTEMAS OPERATIVOS - Prueba 1 - práctica
    cuerpo: nombre de los integrantes
    adjunto: archivo zip comprimido con el siguiente formato
    ⇒ si es un integrante nombre-apellido1.zip
    ⇒ si son dos integrantes nombre-apellido1_nombre-apellido2.zip

La ponderación de las pruebas es la siguiente:

- parte teórica 62% del total de la calificación, **para revisar la parte práctica debe tener por lo menos una calificación 2.5 de 7 en la parte teórica**
- parte práctica 38% del total de la calificación

Se deberá agendar hora con el profesor fuera del horario de clases para revisar en conjunto la parte práctica, si son dos los integrantes deberán asistir los dos, ya que se les harán preguntas y retroalimentación basado en lo que hicieron

# Problema (1) - ponderación 50%

Generar un sistema en c++ capaz de manejar menú con usuarios, autenticación y procesos externos, su sistema deberá:

La apariencia general de su aplicación debe ser:

```
SISTEMA 1 (PID = pid del proceso padre)

############################
respuesta de la ejecución:
—----------------
############################

0 SALIR
1 CREAR USUARIO
2 IMPRIMIR MSG PARA EL USUARIO
3 ORDENAR VECTOR

INGRESE OPCIÓN: x
```

1) Trabajar con archivos de configuración .env, el cual contenga el path de la base de datos de usuario (ej: DB_USERS=/home/lvc/so/prueba1/problema1/db/users.txt)
2) La opción 0 es salir
3) La base de datos de usuario debe ser capaz de manejar el siguiente formato

   userName1;password1
   userName2;password2
   userName3;password3
   .
   .
   .

4) La llamada a ejecución debe ser la siguiente (asumiendo que su aplicación se llama app)

   app -u “lvc” -p “123bnm!” -v “1;5;8;4;9;6”

donde:
-u es el nombre de usuario
-p es la contraseña
-v es un vector de enteros el cual debe ingresar para manipular posteriormente

5) validar la existencia del usuario
6) validar la contraseña del usuario

7) si el userName=admin y pass=admin, ingresó al sistema como administrador y debe tener la opción de crear usuarios, de lo contrario el usuario no debe ver esta opción o no debe tener permiso a utilizarla
8) la opción “1 CREAR USUARIO”, es parte del sistema y permite agregar una línea a la base de datos de usuarios respetando el formato antes señalado, es decir, debe ingresar userName y pass
9) la opción “2 IMPRIMIR MSG PARA EL USUARIO” implica que debe llamar a un proceso externo al sistema, el cual reciba como argumento el nombre de usuario e imprima el mensaje “HOLA COMO ESTAS xxxxx PID”, donde xxxx es el nombre de usuario que recibió el proceso externo y PID del proceso externo
10) la opción “3 ORDENAR VECTOR” implica que debe llamar a un proceso externo al sistema, el cual debe recibir como parámetro el vector de números, ordenarlo, imprimir el resultado ordenado e imprimir el PID del proceso externo

# Problema (2) - ponderación 50%

Generar un sistema en c++ capaz de realizar el conteo de palabras mediante threads, sus sistema deberá:

1) Este sistema no debe realizar ninguna llamada particular ( ej: ./app )

2) Trabajar con archivo .env de configuración en el cual se señale cuatro variables, que son: EXTENSION, PATH_FILES_IN, PATH_FILES_OUT y AMOUNT_THREADS

   ejemplo:

   EXTENSION=txt
   PATH_FILES_IN=/home/lvc/so/prueba1/problema2/files/in
   PATH_FILES_OUT=/home/lvc/so/prueba1/problema2/files/out
   AMOUNT_THREADS=4

   donde:
   EXTENSION, el tipo de archivos a leer
   PATH_FILES_IN, representa la carpeta donde se encuentran N archivos de texto sobre los cuales debe aplicar conteo de palabras
   PATH_FILES_OUT, representa la carpeta de destino, donde por cada archivo sobre el cual realiza conteo de palabras debe establecer un archivo de salida con el mismo nombre

   el resultado del archivo de salida debe ser:
   palabra1; cantidad
   palabra2; cantidad
   palabra3; cantidad
   .
   .
   .

3) Los path PATH_FILES_IN y PATH_FILES_OUT no pueden ser iguales
4) AMOUNT_THREADS, representa la cantidad de threads con los que trabajara su sistema y donde cada threads debe ejecutar un conteo de palabras de un archivo en específico
5) Su sistema debe ser capaz leer el listado de archivos de PATH_FILES_IN, generar un vector con el listado de archivos presentes en el path y luego gatillar la ejecución del conteo por cada archivo asignado a cada thread
6) Una vez finalizado el conteo de cara archivo presente en PATH_FILES_IN su sistema debe imprimir el mensaje

   archivo XXXXXXXX, procesado por el thread ID

   donde:

   XXXXXXXX, representa PATH_FILES_IN/<archivo>.txt
   ID es el identificador del thread que proceso el conteo de palabras

   Ejemplo de salida esperada:

archivo /home/lvc/so/prueba1/problema2/files/in/cuento1.txt, procesado por el thead 0
archivo /home/lvc/so/prueba1/problema2/files/in/cuento10.txt, procesado por el thead 1
archivo /home/lvc/so/prueba1/problema2/files/in/cuento11.txt, procesado por el thead 2
archivo /home/lvc/so/prueba1/problema2/files/in/cuento13.txt, procesado por el thead 3
archivo /home/lvc/so/prueba1/problema2/files/in/cuento15.txt, procesado por el thead 0

.
.
.