TRABAJO PRÁCTICO DE LA ASIGNATURA SISTEMAS OPERATIVOS

El objetivo de este trabajo práctico es familiarizarse con los servicios para la gestión de procesos y archivos que proporciona la interfaz de llamadas al sistema (*system calls*) de POSIX.

El estudiante debe diseñar y codificar un programa que se comporte como un intérprete de comandos, estilo Bash. El programa deberá cumplir las especificaciones descritas en este documento.

Especificaciones mínimas

El intérprete de comandos, o *minishell*, debe utilizar (1) la entrada estándar, o un fichero, para leer la línea de comandos que interpretará y ejecutará; (2) la salida estándar para mostrar el resultado de las órdenes; y (3) la salida de error estándar para notificar los errores que puedan ocurrir.

Mensaje de prompt: En su versión interactiva, la *minishell* mostrará una indicación textual **básica** al usuario que significará que está esperando a que introduzca un comando.

Comandos: La *minishell* admitirá comandos como una lista de tokens separados por blancos (espacios en blanco o carácter tabulador). El primer token especificará el nombre del comando. Los restantes tokens serán los argumentos del comando invocado. La *minishell* leerá y ejecutará comandos hasta que el usuario introduzca la **orden de finalización "salir"**. En ningún caso, la *minishell* se interrumpirá por un comando erróneamente introducido por el usuario. Para ello, cada comando deberá ejecutarse como un proceso hijo del proceso *minishell*.

Scripts: La *minishell* podrá ejecutar órdenes de un fichero de comandos (script), que puede contener solamente comandos del sistema operativo (sin variables, ni estructuras de control, ni comentarios). Si al ejecutar la minishell, el usuario introduce el nombre de un fichero de mandatos como argumento (de la forma: `./minishell fichero`), la *minishell* deberá ejecutar los comandos contenidos en el fichero. Entre los comandos del script puede aparecer la orden "salir", que pondrá fin a la ejecución del script y de la minishell.

Redirección: La entrada o la salida de un comando podrá ser redirigida añadiendo tras él los símbolos '<' o '>'. En caso de error durante las redirecciones, se notificará por la salida de error estándar y se suspenderá la ejecución del comando. Se utilizará la siguiente notación:

- `< archivo` usa archivo como entrada estándar abriéndolo para lectura.
- `> archivo` usa archivo como salida estándar. Si el archivo no existe, se creará; si existe lo truncará (permisos asignados en la creación 0644).

Una tubería: La *minishell* podrá admitir dos comandos separados por el símbolo '|'. La salida estándar del primer comando se conectará por una tubería (pipe) a la entrada estándar del segundo comando. Además, se podrán redireccionar la entrada y la salida estándar de los comandos conectados.

Comando interno cd: La *minishell* reconocerá el comando interno `cd <path>` para cambiar de directorio de trabajo a la ruta indicada en `<path>`.

Especificaciones opcionales

Opcionalmente, se pueden añadir algunas de las siguientes funcionalidades a la *minishell*:

Redirección de errores: Permitir la redirección de la salida de error estándar mediante la siguiente notación:

- `2> archivo` usa archivo como salida de error estándar. Si el archivo no existe se creará, si existe lo truncará (permisos 0644).

Redirección de la salida estándar, añadiendo a fichero: Permitir la redirección de la salida de estándar mediante la siguiente notación:

- `>> archivo` usa archivo como salida estándar. Si el archivo no existe, se creará; si existe, añadirá al final del mismo.

Varios comandos en una línea: Ejecutar varios comandos en una línea separados por '`;`' (tanto en modo interactivo como de fichero script).

Ignorar comentarios en comandos: Si la línea del comando a ejecutar (tanto en modo interactivo, como de fichero script) incluye un carácter de comentario (#), debe ignorarse todo el texto que aparezca detrás de ese símbolo. P.ej.:

- `#echo hola`: la cadena "`echo hola`" es un comentario (la línea no se ejecutará)
- `echo hola # saludo`: solo la cadena "`saludo`" es un comentario (solo se ejecutará `echo hola`)

Mejorar el comando interno `cd`: El comando cd podrá usarse sin especificar `<path>`, o usando el símbolo especial ~ (`cd ~`). En ambos casos se cambiará al directorio home del usuario.

Prompt mejorado: incluir en el prompt el directorio de trabajo actual (solo el directorio; NO el usuario).

ATENCION:
No se valorará ninguna mejora más.

Entregables

1. Fichero, o ficheros, con el **código fuente** (fich**eros cpp).** El programa tiene que estar bien **documentado** con comentarios; bien diseñado (con funciones auxiliares con sentido y suficientes para evitar la repetición de código y facilitar su compresión); y ajustarse a las siguientes **normas de estilo**:
    - Todos los ficheros fuente entregados deben estar identificados con el nombre de su autor.
    - El código debe estar **correctamente sangrado** para facilitar su lectura y la comprensión de la estructura de sus bloques.
    - Como carácter de sangrado debes usar el **espacio en blanco**, **no** el **tabulador**.
    - Puedes sangrar el código con dos o cuatro espacios en blanco.
    - Los nombres de las variables y funciones auxiliares deben ser suficientemente descriptivas.
2. Debe incluirse un fichero en formato markdown con
    - Las instrucciones de construcción del programa a partir del o los ficheros fuente.
    - Un breve manual de uso que describa cómo se inicia y se detiene la ejecución de la minishell, tanto en modo interactivo como de fichero de comandos.

Haz un **fichero ZIP** en el que incluyas los ficheros. **NO ADMITIRÉ FICHEROS RAR, ni 7z… SOLO ZIP**. El nombre del fichero ha de respetar la siguiente convención: <tu_nombre_completo>_<tu_DNI>_TR.zip.

Evaluación

El trabajo debe ser original. Se usarán herramientas automáticas para la detección de plagios. En caso de plagio, la calificación será de suspenso, lo que supondrá el suspenso en la calificación final de la asignatura.

Fecha tope de entrega convocatoria ordinaria: **14 de diciembre de 2025** (domingo) a las 24 h.

La evaluación del programa entregado se regirá por los siguientes criterios:

Se evaluará la corrección, robustez y calidad de la escritura del código (normas de estilo), tanto de las especificaciones mínimas como opcionales.
La calificación se ajustará las siguientes reglas, obteniéndose el máximo de puntuación indicada en cada caso si el funcionamiento es totalmente correcto y robusto.

- Especificaciones mínimas: máximo 6 ptos.
- Especificaciones opcionales: máximo 4 ptos. Se valorarán una vez obtenido un mínimo de 5 puntos (sobre 6) en las especificaciones mínimas. Puedes obtener puntos implementando las siguientes mejoras:
    - Redirección de errores: máximo 0,625 ptos.
    - Redireccionar la salida estándar, añadiendo a fichero: máximo 0,625 ptos.
    - Varios comandos en una línea: máximo 0,75 ptos.
    - Ignorar comentarios en comandos: máximo 0,75 ptos.
    - Prompt incluyendo directorio de trabajo: máximo 0,5 ptos.
    - Mejora en el comando `cd`: máximo 0,75 ptos.
- Consideraciones especiales:
    - Si no se respetan las normas de estilo indicadas en el apartado de "Entregables" la nota final será disminuida en un máximo de 1 punto.
    - No se valorarán otras mejoras, concretamente todas aquellas funcionalidades añadidas con el único propósito de decorar la ejecución del programa o modificar su aspecto visual: letras de colores, efectos gráficos…

Evaluación presencial

De forma aleatoria o discrecional el profesor podrá convocar a una entrevista personal a estudiantes que hayan entregado el trabajo para recibir explicaciones sobre la forma en la que lo han realizado. Durante esa entrevista, las explicaciones aportadas deben ser claras y completas. De no serlo, la calificación del trabajo puede verse afectada negativamente, pudiéndose incluso producir el suspenso.