**Taller #2**

**Teoría**

1. Explique las etapas de creación de un proceso en Windows (CreateProcess).

El método CreateProcess es una función en el entorno de programación de Windows que se utiliza para crear un nuevo proceso o programa ejecutable. Esta función es ampliamente utilizada por los desarrolladores para iniciar nuevas aplicaciones o ejecutar programas en el sistema operativo Windows.

Etapas:

* **Preparación de la estructura STARTUPINFO**: Esta estructura contiene información sobre cómo se debe mostrar la ventana principal del nuevo proceso, cómo se manejarán las ventanas secundarias, etc.
* **Preparación de la estructura PROCESS\_INFORMATION**: Se almacenarán los detalles sobre el nuevo proceso creado, como su identificador (ID) y el identificador del hilo principal.
* **Llamada a CreateProcess:** Esta función toma varios parámetros, incluyendo el nombre del ejecutable o archivo a ejecutar, la línea de comandos si es necesario, y punteros a las estructuras STARTUPINFO y PROCESS\_INFORMATION.
* **Creación del proceso:** Windows crea un nuevo proceso en el sistema. Esto incluye la asignación de recursos, la carga de bibliotecas y la configuración de entorno necesario para la ejecución del nuevo programa.
* **Carga de bibliotecas:** Si el programa que se está ejecutando depende de bibliotecas dinámicas (DLL), estas bibliotecas se cargarán en el proceso recién creado.
* **Inicio de ejecución:** Una vez que el proceso se ha creado y las bibliotecas se han cargado, el punto de entrada del programa (generalmente la función main en C/C++) se ejecutará, iniciando la ejecución del programa.
* **Finalización y cierre:** Cuando el programa finalice su ejecución, el proceso saldrá de forma controlada o abrupta, dependiendo de cómo se haya implementado el código. Windows también se encargará de liberar los recursos asociados al proceso, como la memoria asignada y los descriptores de archivos abiertos.
* **Manejo de errores:** Durante todo el proceso, pueden ocurrir diversos tipos de errores, como incapacidad para encontrar el archivo ejecutable, problemas de permisos, etc. Es importante manejar estos errores adecuadamente en el código para proporcionar una experiencia de usuario más robusta.

1. ¿Cuáles son las variables necesarias que se deben de guardar cuando se quiere implementar un cambio de contexto?

El cambio de contexto es el proceso mediante el cual un sistema operativo permite que múltiples procesos compartan el tiempo de ejecución en un solo procesador. Durante un cambio de contexto, el sistema operativo guarda el estado actual de un proceso en ejecución y carga el estado de otro proceso en la CPU para que continúe su ejecución.

* **Registro de la CPU:** Los valores actuales de los registros de la CPU, que incluyen registros generales, registros de control y registros de segmento, deben ser guardados.
* **Puntero de pila (Stack Pointer):** El puntero de pila indica la ubicación actual en la pila del proceso. Debe guardarse para asegurarse de que el proceso pueda continuar su ejecución desde el mismo punto cuando se restaure.
* **Estado de los registros de punto flotante:** Si el procesador admite instrucciones de punto flotante, el estado de los registros de punto flotante debe guardarse y restaurarse adecuadamente para mantener la integridad de los cálculos.
* **Estado de los registros de control:** Los registros de control y estado del procesador también deben guardarse para que el proceso pueda continuar ejecutándose como si no hubiera habido un cambio de contexto.
* **Estado de la memoria:** Cualquier área de memoria que sea exclusiva del proceso, como el código, los datos y la pila, debe estar intacta.
* **Información del planificador de procesos:** El sistema operativo necesita saber qué proceso se reanudará a continuación y en qué punto. Esto implica tener una estructura de datos que almacene información sobre todos los procesos en espera y sus prioridades, de manera que se pueda tomar la decisión adecuada al momento de realizar el cambio de contexto.

1. Para que sirve el comando ps y top en un entorno de Linux.

**Comando ps:** El comando ps (Process Status) se utiliza para mostrar información sobre los procesos que se están ejecutando en el sistema en el momento de ejecutar el comando.

* ps aux: Muestra una lista detallada de todos los procesos del sistema, incluyendo información sobre el usuario, el ID del proceso (PID), el uso de CPU, la memoria utilizada, etc.

**Comando top:** El comando top es una herramienta interactiva que muestra una vista en tiempo real de los procesos en ejecución y la utilización del sistema. Cuando ejecutas top, obtienes una ventana que se actualiza periódicamente para mostrar información actualizada sobre la CPU, la memoria y los procesos.

1. Investigue los posibles estados de un proceso en un entorno de Linux y cómo se representan.

En un entorno de Linux, los procesos pueden encontrarse en varios estados diferentes, que indican en qué punto de su ciclo de vida se encuentran. Los posibles estados de un proceso en Linux son los siguientes:

**Ejecución (Running):** El proceso está siendo ejecutado en la CPU en este momento.

**Listo (Ready):** El proceso está listo para ejecutarse, pero está esperando su turno para ser asignado a la CPU.

**Bloqueado (Blocked):** El proceso está esperando algún evento o recurso externo, como entrada/salida (E/S) o una señal, y no puede continuar.

**Suspendido (Suspended)**: El proceso está detenido temporalmente y no se ejecuta.

**Terminado (Terminated):** El proceso ha completado su ejecución y ha sido finalizado.

Estos estados se representan mediante códigos de estado en la tabla de procesos del sistema operativo.

R: Running (Ejecución)

D: Uninterruptible Sleep (Bloqueado)

S: Interruptible Sleep (Listo)

T: Stopped (Suspendido)

Z: Zombie (Terminado, pero la entrada de la tabla de procesos aún está presente)

X: Dead (Terminado)

1. ¿Cómo se podría implementar un cambio de contexto por hardware y no por software? Realice un esquema de arquitectura con su propuesta.

Un cambio de contexto por hardware implica que el proceso de cambio de un proceso en ejecución a otro se realiza principalmente mediante circuitos y mecanismos internos del procesador, en lugar de depender completamente del software del sistema operativo.

**Arquitectura de Cambio de Contexto por Hardware:**

**Registro de Contexto:** El procesador tendría un conjunto de registros de contexto dedicados para cada proceso en ejecución. Estos registros almacenarían información crucial del proceso actual, como registros de propósito general, registros de punto flotante, punteros de pila, etc.

**Controlador de Cambio de Contexto:** Habría un controlador específico dentro del procesador encargado de gestionar los cambios de contexto. Este controlador podría estar vinculado a temporizadores, interrupciones u otros eventos que podrían desencadenar un cambio de contexto.

**Tabla de Descriptores de Procesos:** Una tabla en memoria o en registros especiales podría contener los descriptores de los procesos en ejecución.

**Señalización de Cambio de Contexto:** Cuando el controlador de cambio de contexto detecta que es necesario realizar un cambio, podría enviar señales internas al procesador para indicar que el estado actual del proceso debe guardarse y el siguiente proceso debe cargarse.

**Guardado y Carga Automática de Contexto:** El controlador activaría el mecanismo de guardado automático del contexto del proceso actual en los registros de contexto dedicados. Al mismo tiempo, cargaría el contexto del siguiente proceso desde los registros de contexto correspondientes.

**Restauración de Punteros y Registros**: Los punteros de pila y otros registros cruciales se restaurarían durante este proceso para que el nuevo proceso pueda continuar exactamente donde se detuvo.

**Reanudación de Ejecución:** Una vez que el nuevo contexto esté cargado y listo, el procesador reanudaría la ejecución del nuevo proceso a partir del punto en que se detuvo.

**Control de Interrupciones:** Durante el cambio de contexto, el procesador podría estar configurado para no permitir interrupciones, evitando así interferencias mientras se realiza el cambio.

Este es solo un esquema conceptual y simplificado. En una arquitectura real, habría más detalles y consideraciones, como la gestión de la caché, la administración de recursos compartidos, el manejo de excepciones y más. Un cambio de contexto por hardware es más eficiente en términos de velocidad de cambio y podría ser más adecuado para sistemas en los que se requiere una multitarea muy rápida y eficiente.

**Procesos en Linux**

1. Conéctese a su máquina virtual por medio de SSH. Acceda mediante dos conexiones, es decir, dos consolas.
2. Ejecute el comando: ps -aux (obtenga la captura de los últimos ítem) explique cual es el significado de aux.

Estos son argumentos o opciones que se pasan al comando "ps" para personalizar su salida y comportamiento. Aquí está el desglose de cada parte de los argumentos:

"-a": Este argumento muestra información sobre todos los procesos en el sistema, no solo los asociados con el terminal actual. Incluso muestra procesos de otros usuarios.

"-u": Este argumento muestra información detallada sobre los procesos, incluyendo el nombre del usuario que inició cada proceso.

"-x": Este argumento muestra procesos que no están asociados con un terminal. Incluye procesos de fondo (daemons) y otros procesos que no tienen una interfaz de usuario directa.

A screen shot of a computer

Description automatically generated

1. Investigue cada uno de los datos de los procesos del punto anterior (PID, VSZ ...).

**USER:** El nombre del usuario que inició el proceso.

**PID:** El Identificador de Proceso (Process ID) es un número único asignado a cada proceso en el sistema. Se utiliza para identificar y controlar procesos.

**%CPU:** El porcentaje de uso de la CPU que está siendo utilizado por el proceso en el momento en que se ejecutó el comando "ps". Este valor puede variar con el tiempo.

**%MEM:** El porcentaje de memoria RAM física en uso por el proceso en el momento en que se ejecutó el comando "ps".

**VSZ:** Virtual Set Size, es la cantidad de memoria virtual en kilobytes que el proceso está utilizando. Incluye tanto la memoria física como la memoria intercambiada al disco.

**RSS:** Resident Set Size, es la cantidad de memoria RAM física en kilobytes que el proceso está utilizando en ese momento. Esta es la memoria que realmente está en uso por el proceso en la RAM.

**TTY:** El nombre de la terminal asociada con el proceso. Puede ser una terminal física o una terminal virtual.

**STAT:** El estado del proceso, que puede incluir letras como "R" (Ejecución), "S" (Espera), "D" (Durmiendo), "Z" (Zombie), entre otros.

**START:** La hora de inicio del proceso.

**TIME:** El tiempo total de CPU utilizado por el proceso desde que se inició.

**COMMAND:** El comando o programa que inició el proceso, junto con sus argumentos.

1. Busque el comando que retorna los procesos propios de un usuario y tome la captura de pantalla del que posee mayor tiempo en el procesador.

A screenshot of a computer

Description automatically generated

1. Ejecute el comando: top en la primera consola. ¿Para qué sirve?

El comando top se utiliza para monitorear en tiempo real la actividad del sistema y la utilización de recursos, como la CPU, la memoria RAM y los procesos en un sistema Unix o Linux. Al ejecutar top en la terminal, se muestra una lista de información en curso que se actualiza periódicamente, lo que permite a los usuarios obtener una visión general de cómo se está comportando el sistema en ese momento.

A screenshot of a computer

Description automatically generated

1. Ejecute el comando (5 veces en la segunda consola): cat /dev/zero > /dev/null &

A screen shot of a computer screen

Description automatically generated

1. Ejecute el comando top nuevamente en la primera consola. ¿Qué observa con respecto al top anterior?

Que los ID de los procesos en el comando anterior se están ejecutando con el comando “cat”

A screenshot of a computer

Description automatically generated

1. ¿Qué significa los valores de cada uno de los parámetros de los procesos creados (PR, NI, VIRT ...)?

**PID (Process ID):** El ID de proceso es un número único asignado a cada proceso en el sistema.

**USER:** El nombre de usuario que inició el proceso.

**PR (Priority):** La prioridad del proceso. Es un valor numérico que representa la prioridad del proceso en la cola de ejecución. Los valores más bajos indican una prioridad más alta.

**NI (Nice Value):** El valor "nice" del proceso.

**VIRT (Virtual Memory):** La cantidad de memoria virtual utilizada por el proceso en kilobytes. Esto incluye la memoria física y la memoria intercambiada al disco.

**RES (Resident Memory):** La cantidad de memoria RAM física en kilobytes que el proceso está utilizando en ese momento.

**SHR (Shared Memory):** La cantidad de memoria compartida en kilobytes utilizada por el proceso.

**S (%CPU):** El porcentaje de uso de CPU por parte del proceso en ese momento.

**MEM (%MEM):** El porcentaje de memoria física en uso por el proceso en ese momento en relación con la memoria total del sistema.

**TIME+:** El tiempo total de CPU consumido por el proceso desde su inicio.

**COMMAND:** El nombre del comando o programa que inició el proceso, junto con sus argumentos.

1. Note que todos los procesos creados tienen una prioridad similar ¿Por qué sucede esto?

El comando cat /dev/zero > /dev/null & no modifica explícitamente la prioridad de los procesos que crea, por lo que heredan la prioridad predeterminada de su shell. Como resultado, estos procesos generalmente tendrán una prioridad similar porque se ejecutan con los mismos valores "nice" predeterminados.

1. ¿Por qué el parámetro “ Time“ aumenta paulatinamente?

El parámetro "Time" en la salida del comando top representa el tiempo total de CPU consumido por un proceso desde su inicio. Este valor aumenta paulatinamente a medida que el proceso continúa ejecutándose porque está registrando el tiempo de CPU que ha utilizado desde que comenzó su ejecución.

1. Obtenga un identificador de alguno de los procesos creados anteriormente

A black and white screen with numbers

Description automatically generated

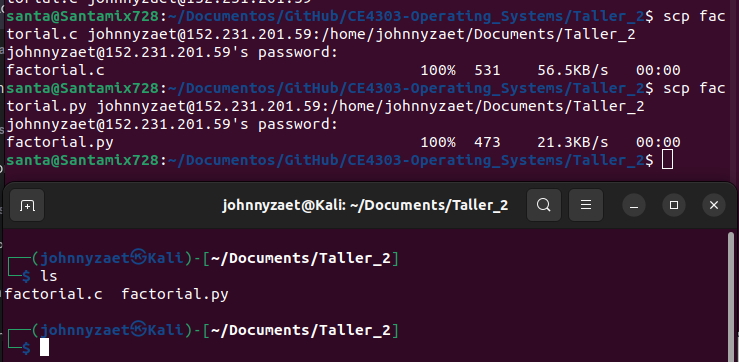
1. Aumente la prioridad de dicho proceso con el comando: renice -n 10 PID
2. Inicie un proceso con prioridad alta con el siguiente comando: nice -n -10 cat /dev/zero > /dev/null &
3. Ejecute el comando top nuevamente.

A screenshot of a computer

Description automatically generated

**Creación de procesos con parámetros establecidos por el usuario**

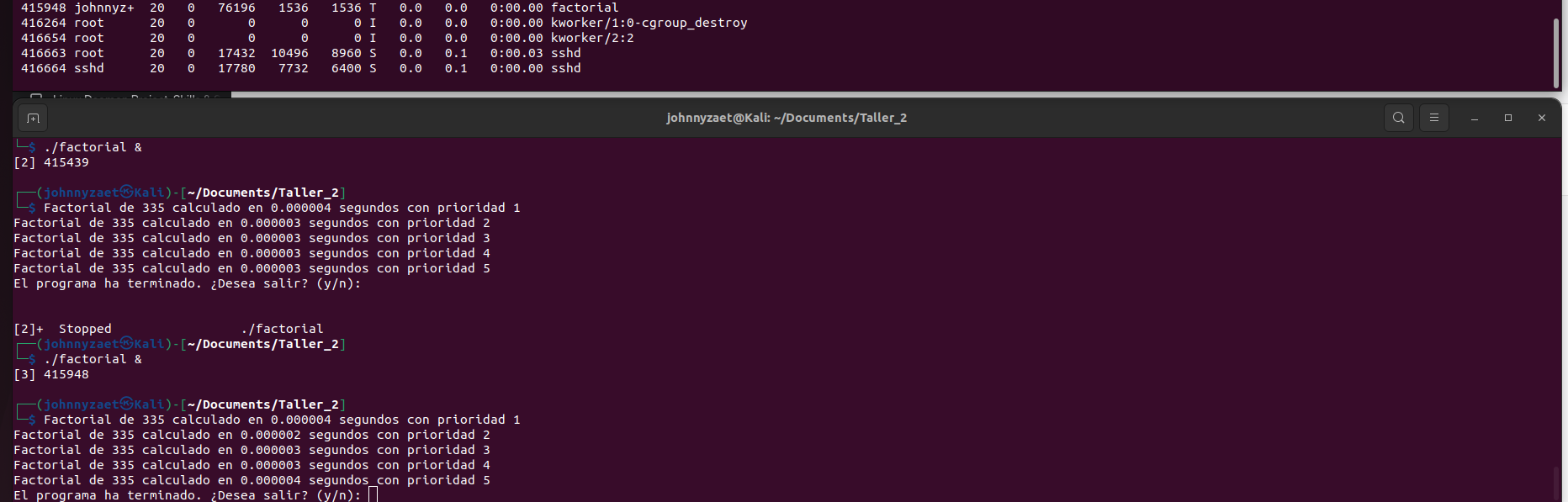
Se realizaron ambos programas y se pasaron a la maquina virtual mediante ssh.



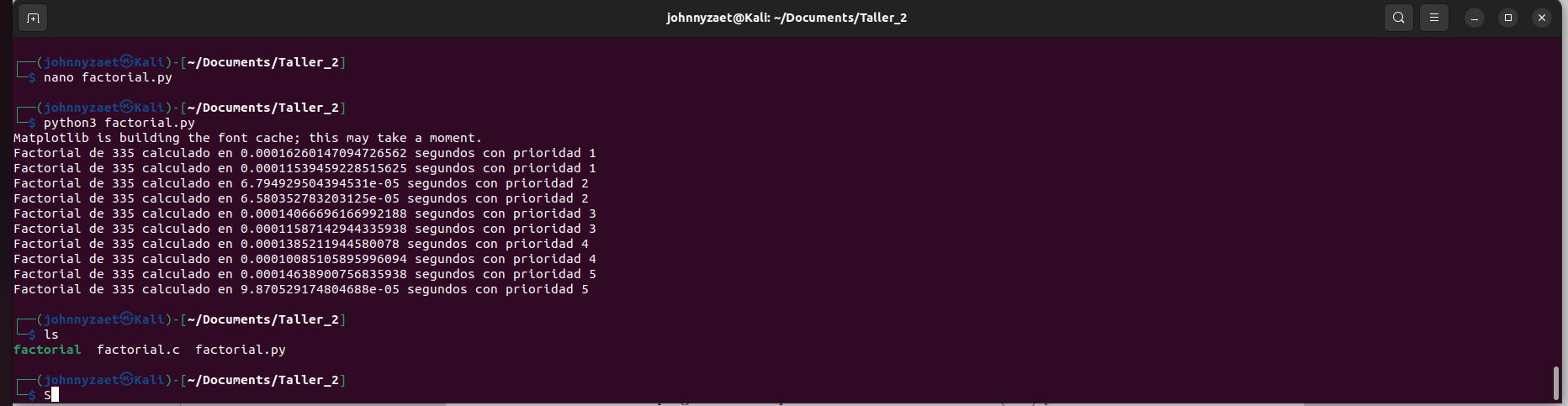
Texto

Descripción generada automáticamente





Se ejecutaron 5 veces con distintas prioridades



Texto

Descripción generada automáticamente

En el caso de ambas graficas podemos observar que poseen el comportamiento que se esperaba, dando valores en condiciones similares, para el caso de la gráfica de Python se realizó dos valores por cada prioridad, por lo que los valores que mas coinciden con la comparación del C son los valores inferiores, lo que vendría siendo la primera iteración de la tabla que se observa en la imagen de la ejecución del programa de Python.

Se realizó la eliminación del proceso mediante el comando kill.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**Hilos en Linux**

Forma, Rectángulo

Descripción generada automáticamente

La mejora en el rendimiento al aumentar la cantidad de hilos no necesariamente será lineal en este caso, y su comportamiento dependerá de varios factores, como la naturaleza del problema, el número de núcleos de CPU disponibles y el tamaño del archivo. A continuación, se describen algunas consideraciones sobre por qué la mejora puede no ser lineal:

* Sobrecarga de concurrencia: A medida que se agregan más hilos, puede haber una sobrecarga en la administración de hilos y en la sincronización entre ellos. Esta sobrecarga puede reducir la eficiencia y, en algunos casos, incluso puede llevar a un rendimiento inferior al usar demasiados hilos.
* Límites de hardware: El número de núcleos de CPU disponibles en el hardware limita la capacidad de ejecución concurrente. Si tienes un procesador de cuatro núcleos, por ejemplo, es poco probable que obtengas una mejora lineal más allá de cuatro hilos, ya que los hilos adicionales tendrán que competir por el tiempo de CPU.
* Naturaleza secuencial: En este caso, estás realizando una operación secuencial al leer el archivo línea por línea. Aunque puedes dividir el archivo entre varios hilos, la naturaleza secuencial de la búsqueda de palabras en el archivo limita el grado de paralelismo que se puede lograr.
* Sincronización: La sincronización entre hilos, como la espera de que todos los hilos completen su tarea, puede agregar cierta latencia al proceso, lo que puede afectar la mejora en el rendimiento.
* Tamaño del archivo: El tamaño del archivo y la cantidad de datos a procesar pueden influir en el rendimiento. Para archivos pequeños, la ganancia en el rendimiento al usar múltiples hilos puede ser limitada.