Modificaciones al sistema operativo XV6

Integrantes:

- Vladimir Roger Ticona Mamani 2023-119063
- Jorge Enrique Obando Huallpa 2017-130045
- Edu Rubinho Puma Ccama 2023-119053

Introduccion

- **¿Qué es XV6?**
  - Sistema operativo educativo desarrollado por el MIT
- **¿Por qué XV6?**
  - El tamaño de código es manejable, es de codigo abierto y educativo para enseñanza
- **Conceptos Fundamentales:**
  - System Calls(Syscalls)
  - Modo Usuario vs Modo Kernel
  - Gestión d

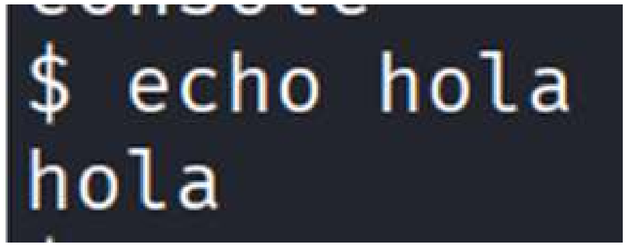
Objetivos

- **Objetivo general**
  - Extender la funcionalidad del sistema operativo XV6 mediante la implementación de mecanismos de instrumentación, monitoreo y análisis para comprender su funcionamiento interno.
- **Objetivos especificos:**
  - Instrumentación de Syscalls
  - Comandos de Monitoreo
  - Contador de Invocaciones

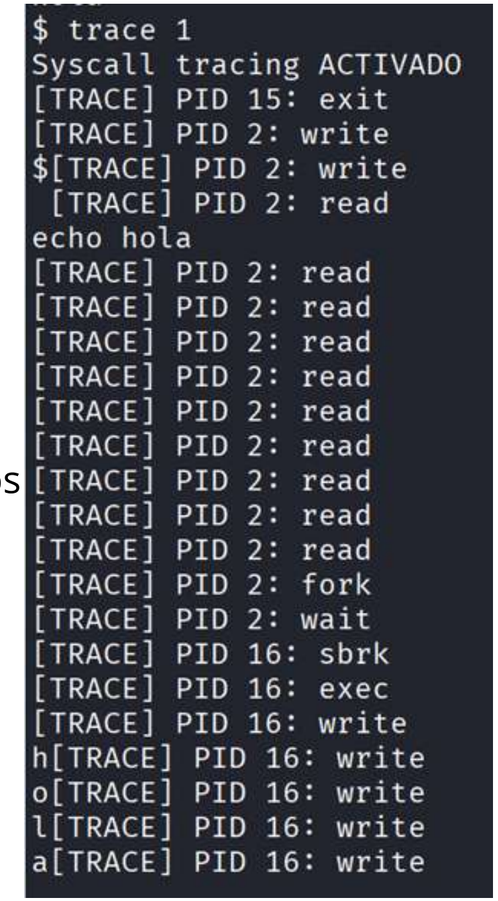
ENTREGABLE 1

- **Instrumentacion de syscall**
- **Activación/desactivación mediante comando `trace`**
- **Muestra nombre de la syscall**
- **Indica el PID del proceso**
- **Se ejecuta de forma transparente**

Captura 1:
Se prueba el comando antes de ver los resultados



Captura 2:
Se prueba el comando para ver los resultados



ENTREGABLE 2

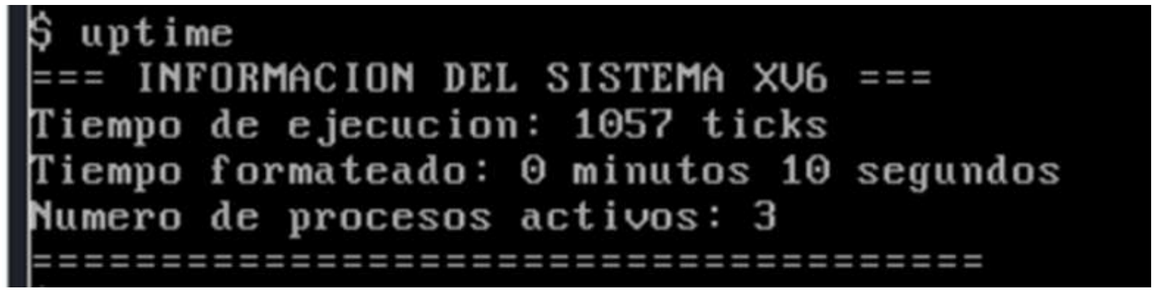
- **Comandos para ver la informacion del sistema**

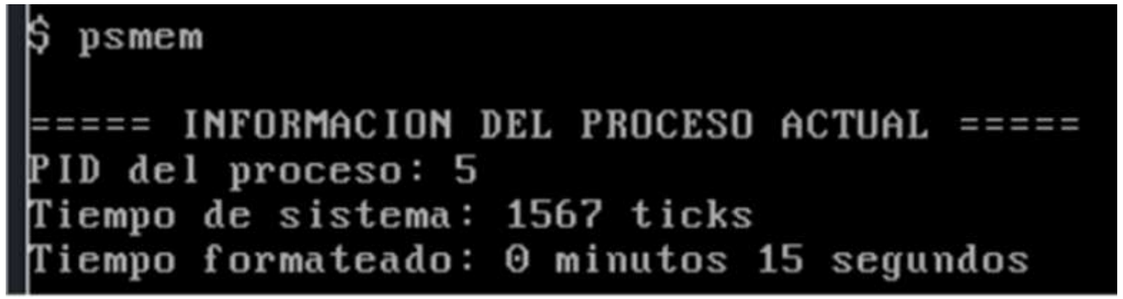
Agregamos dos nuevas syscalls en el kernel:

- numprocs: que cuenta cuantos procesos estan activos
- getmem: que te dice cuanta memoria tiene asignada el proceso actual

Luego creamos dos programas de usuario:

- uptime.c: Este programa te muestra cuanto tiempo lleva el sistema corriendo. Obtiene los ticks de reloj del sistema (XV6 cuenta muy rapido, unos 100 ticks por segundo), los convierte a segundos y minutos, y los muestra en pantalla junto con el numero de procesos activos.
- psmem.c: Este programa muestra informacion del proceso que lo ejecuta. Te dice el PID, cuanta memoria tiene asignada (en bytes y kilobytes), y cuanto tiempo lleva el sistema funcionando



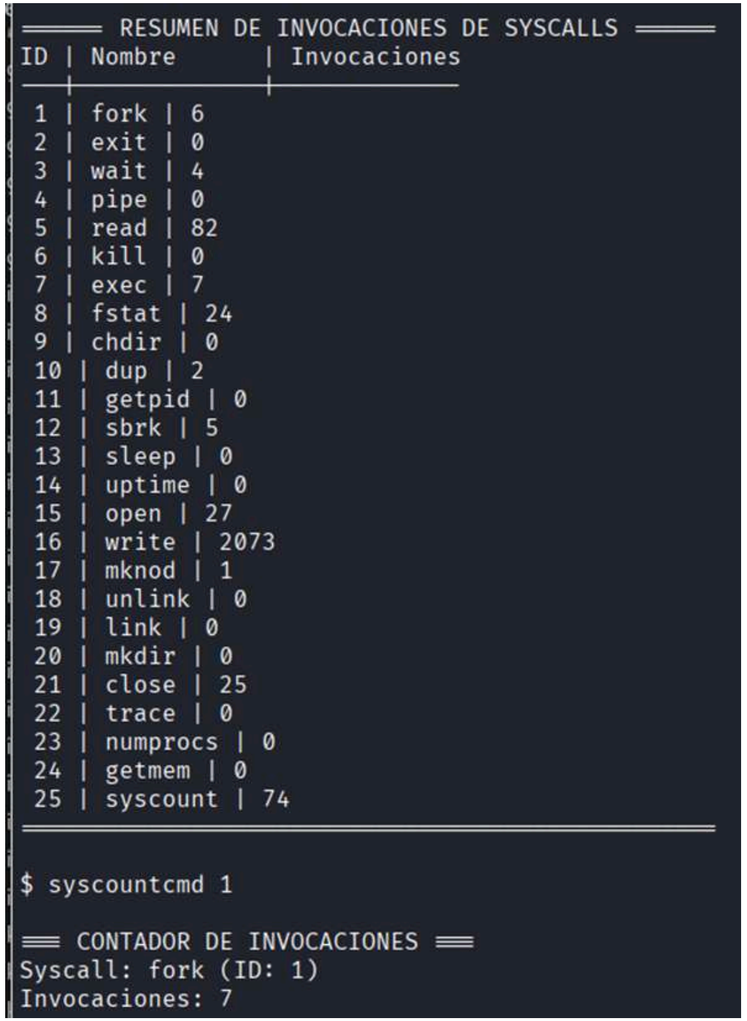


ENTREGABLE 3

- **CONTAR CUANTAS VECES SE USA CADA SYSCALL**

  - Agregamos un arreglo en el kernel llamado syscall_count con 26 espacios (uno para cada syscall). Lo inicializamos en ceros.
  - Implementamos una nueva syscall llamada "syscount" que te permitia consultar estos contadores desde un programa de usuario.
  - Finalmente, creamos un programa llamado syscountcmd.c que te mostraba estos contadores. Si lo ejecutabas sin argumentos, veias una tabla con todas las syscalls y sus contadores. Si escribias "syscountcmd 5", te mostraba solo el contador de la syscall numero 5 (que es read).

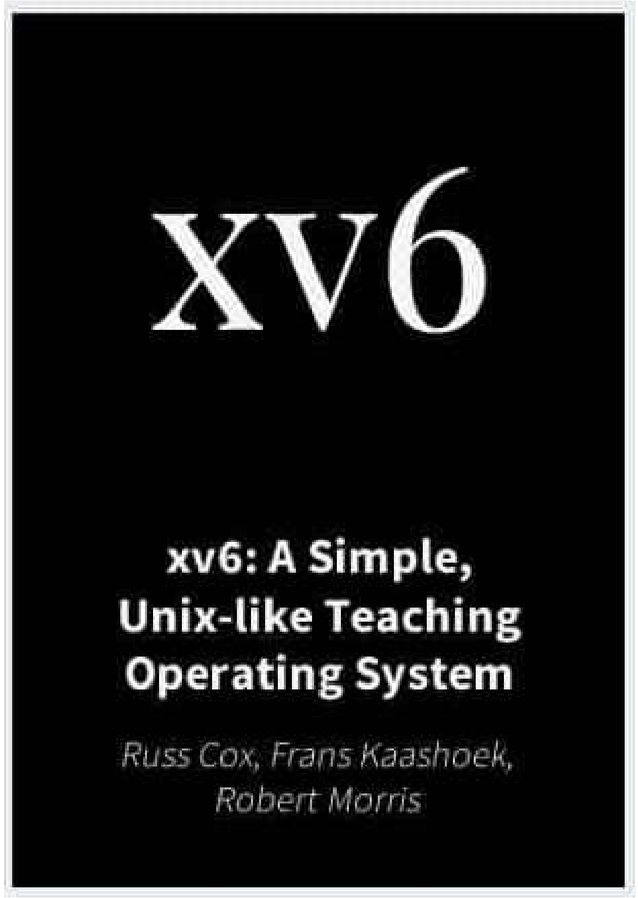
Cuando ejecutabas "ls", veias que fork subia en 2, exec subia en 2, read subia en muchos, write subia mucho (porque tiene que mostrar el resultado en pantalla), y close subia varios. Esto nos ayudo a visualizar exactamente que hace XV6 cuando ejecutas un comando.



Demostracion de los comandos

CONCLUSIONES

El proyecto nos permitió comprender de manera práctica cómo funcionan los sistemas operativos: vimos que las syscalls son la puerta entre el modo usuario y el kernel, que cada comando se ejecuta en un proceso separado mediante fork, exec y wait, y que la memoria de cada proceso está completamente aislada. Observamos además la gran cantidad de actividad que ocurre incluso con comandos simples y entendimos la importancia de validar la seguridad y el correcto funcionamiento de estas operaciones. La experiencia nos mostró que los sistemas operativos no son magia, sino código que, al comprender sus conceptos básicos, podemos modificar y extender, consolidando así nuestra transición de la teoría a la práctica.



¡Muchas gracias!