UVG Esteban Del Valle

Sistemas Operativos 18221

Laboratorio Sección 20

Llamadas al sistema y Kernel

**Ejercicio 1**

Compile el primer programa y ejecútelo varias veces. Responda: ¿por qué aparecen números diferentes cada vez?

Porque getpid imprime el proceso que se asigna a esta llamada al sistema y cuando se vuelve a llamar el numero incremente dependiendo de la cantidad de veces que se ejecute.

Proceda a compilar el segundo programa y ejecútelo una vez. ¿Por qué aparecen dos números distintos a pesar de que estamos ejecutando un único programa?

Esto es dado a que la llamada fork crea un proceso hijo. Un numero corresponde al proceso padre y otro corresponde al proceso hijo

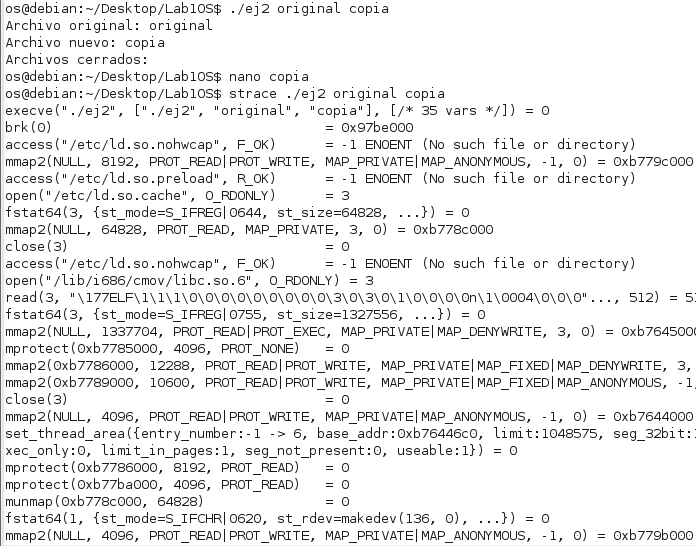
¿Por qué el primer y el segundo números son iguales?

Cuando se realiza una llamada al sistema fork la instrucción siguiente se hace de forma simultánea inmediatamente.

En la terminal, ejecute el comando top (que despliega el top de procesos en cuanto a consumo de CPU) y note cuál es el primer proceso en la lista (con identificador 1). ¿Para qué sirve este proceso?

El proceso con id 1 es systemd y es el primer proceso que genera el SO para poder llevar a cabo todos los demás procesos. Este proceso contiene muchas herramientas que lo ayudan a interactuar con el kernel.

**Ejercicio 2**



Observe el resultado desplegado. ¿Por qué la primera llamada que aparece es execve?

Execve es el proceso que ejecuta el programa referido. Si no se lleva a cabo este proceso, el programa no se ejecutaría. De este modo el programa puede empezar.

Ubique las llamadas de sistema realizadas por usted. ¿Qué significan los resultados (números que están luego del signo ‘=’)?

Los números después del signo = son los valores retornados por la función. También simbolizan si hubo algún error en el proceso y dependiendo si regresa 0 u 1 se puede saber.

¿Por qué entre las llamadas realizadas por usted hay un read vacío?

El read vacio simboliza que se llegó el end of file es decir el final del archivo y ya no hay nada mas que copiar.

Identifique tres servicios distintos provistos por el sistema operativo en este strace. Liste y explique brevemente las llamadas a sistema que corresponden a los servicios identificados (puede incluir read, write, open o close que el sistema haga por usted, no los que usted haya producido directamente con su programa).

Mmap2: esta llamada crea un nuevo mapeo en la dirección virtual del espacio del proceso que se esta llamando. En este caso se utiliza de la siguiente manera. mmap2(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xb7

Mprotect: esta llanada cambia las protecciones de acceso para las llamadas de las paginas de memoria conteniendo cualquier parte de la dirección en un intervalo. En este caso se usa así. mprotect(0xb777d000, 8192, PROT\_READ) = 0

Fstat: esta llamada regresa información acerca de un archivo. No se requieren permisos y y muestra ka información del archivo especificado por el path y el archivo debe de estar descrito por el descriptor fd. Se utiliza de esta manera. fstat64(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0

**Ejercicio 3**

¿Qué ha modificado aquí, la interfaz de llamadas de sistema o el API? Justifique su respuesta.

Se modifica la interfaz de llamadas porque el API ya es fijo y no se puede cambiar. Se agrego una llamda al sistema dentro de la interfaz de llamadas.

¿Por qué usamos el número de nuestra llamada de sistema en lugar de su nombre?

Esto es porque la manera de invocar llamadas al sistema. Mediante este numero que es su ID. No es posible invocar a una llamada mediante su nombre. Solo mediante su ID.

¿Por qué las llamadas de sistema existentes como read o fork se pueden llamar por nombre?

Esto es porque estas llamadas tienen unos wrappers que permiten que estas sean llamada mediante estos alias. Glibc tiene todas las llamadas pre exsitentes que se pueden llamar por su nombre.

Incluya entre sus respuestas una captura de pantalla con el resultado de la ejecución de su llamada a sistema.

