Michael Chan - 18562
Laboratorio 1 Sistemas Operativos

Ejercicio 1

Compile el primer programa y ejecútelo varias veces. Responda:¿por qué aparecen números diferentes cada vez?
Debido a que cada vez que corremos el programa lo hacemos en un nuevo proceso y getpid lo que nos devuelve es el número del proceso.

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?
Aparecen dos números distintos, porque el fork hace que se separe el programa y se corran en dos procesos distintos.

¿Por qué el primer y el segundo números son iguales?
Porque se ejecutaron en la misma línea de procesamiento.

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?
Es el proceso padre que se ejecuta al prender la computadora, de él surgen todos los demás procesos.

Ejercicio 2

Observe el resultado desplegado. ¿Por qué la primera llamada que aparece es `execve`?
Debido a que representa el ejecutar el programa solicitado.

Ubique las llamadas de sistema realizadas por usted. ¿Qué significan los resultados (números que están luego del signo '=')?
Representan si están en el proceso padre o son hijos que surgieron mientras corría el programa.

¿Por qué entre las llamadas realizadas por usted hay un `read` vacío?
Siempre al acabar de leer un archivo nos encontraremos con un read que representa el final de la lectura del archivo.

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).
mmap representan instrucciones de manejo de memoria.
El primer read se encarga de leer la metadata de nuestro programa.
arch_prctl nos define una arquitectura para trabajar.

Ejercicio 3

¿Qué ha modificado aquí,la interfaz de llamadas de sistema o el API? Justifique su respuesta.
Tal como lo comparamos con una interfaz de Java, lo que hicimos en este caso ha sido agregar una función a nuestra interfaz, que podemos llamar por medio del número que le asignamos, en este caso el 345, y lo que hará es sumar el número que nosotros hayamos definido.

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

Debido a la gran cantidad de llamadas de sistema que existen, es más fácil organizarlas por medio del número.

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

Debido a que son llamadas de sistema bastante comunes.

***Foto de llama de sistema en carpeta de capturas.**