Sistemas Operativos

Taller de Syscalls

Grupo 20

Daniel Grosso	694/08	dgrosso@gmail.com
Mariano De Sousa Bispo	389/08	marian_sabianaa@hotmail.com

Septiembre 2010

1. Explicación de `mister`

El ejecutable `mister` al inicio genera dos *pipes* (que llamaremos `p1` y `p2`) y un proceso hijo. El comportamiento del proceso padre y el proceso hijo se describe a continuación:

Proceso padre

- Establece `p1` como *pipe* de sólo escritura y `p2` como *pipe* de sólo lectura.
- Escribe la palabra "vaca" en `p1` y espera a que el proceso hijo envíe una respuesta a través de `p2`.
- Repite el mismo procedimiento con dos frases distintas.
- Escribe la palabra "chau" en `p1` y espera a que el proceso hijo envíe una respuesta a través de `p2`.
- Cierra los *pipes*.
- Termina.

Proceso hijo

- Establece `p1` como *pipe* de sólo lectura y `p2` como *pipe* de sólo escritura.
- Espera a recibir un *string* en `p1` y escribe en `p2` la longitud del *string*.
- Repite el paso anterior hasta que el padre cierra los *pipes*.
- Cierra los *pipes*.
- Termina.

2. Programa `nofork`

El programa `nofork` ejecuta el comando pasado por parámetro, no permitiéndole a este ejecutar ninguna de las *syscalls* prohibidas: `fork`, `clone`. Para ello, crea un proceso hijo mediante la instrucción `fork`, quien llama a la *syscall* `ptrace` con parámetro `PTRACE_TRACEME` para permitirle al padre rastrear la ejecución[1]. Luego reemplaza al proceso actual (hijo) con el comando pasado por parámetro mediante la instrucción `execvp`. Por otra parte, el padre espera a que se produzca un cambio de estado en el proceso hijo vía `wait` y, mediante una llamada a `ptrace` con parámetro `PTRACE_PEEKUSER`, obtiene el valor del registro `EAX`, que es el número de *syscall* que ejecutó el proceso hijo. Si la *syscall* era `fork` o `clone`, mata al proceso con una llamada a `ptrace` con parámetro `PTRACE_KILL`, imprime *Syscall prohibida* en `stdout` y termina el programa. En caso contrario, se sigue la ejecución del hijo hasta la próxima *syscall* a través de una llamada a `ptrace` con parámetro `PTRACE_SYSCALL`. Si ninguna de las *syscalls* prohibidas fue ejecutada, el proceso hijo se ejecuta con normalidad y, al terminar de ejecutarse, termina el programa.

[1] `PTRACE_TRACEME` hace que cualquier señal (salvo `SIGKILL`) enviada al proceso lo detenga y notifica al padre vía `wait`.