En qué sistema operativo trabajaron

En una máquina virtual corriendo Linux con la distribución wifislax 4.1 con anfitrión Windows 7.

¿Qué programa emplearon para obtener la traza?

Strace

¿Qué programa objetivo trazaron?

Date

¿Por qué eligieron este programa?

Por ser aparentemente un comando sencillo de explicar

Sus observaciones / resultados

Las explicaciones de los comandos las obtuve de:

http://man7.org/linux/man-pages/dir_all_alphabetic.html

Solo el inicio y final están comentados.

```
wifislax64 ~ # strace date
execve("/usr/bin/date", ["date"], [/* 67 vars */]) = 0
```

execve ejecuta el programa o script, a este hay que darle una ubicación el proceso a ejecutar y después los argumentos o variables que usara ese proceso. El cero es por no haber ningún error.

```
brk(NULL)                               = 0x1502000
```

brk aumenta la memoria del proceso, cuando se le da null solo devuelve la dirección final de la memoria del proceso.

```
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fabccb78000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso es null, después la longitud en bytes, seguido del tipo de protección de la memoria en este caso de escritura y lectura, posteriormente crea una copia para escritura e indica que no está respaldada por un archivo y se inicializa en cero, lleva un -1 en descriptor de archivo por usarse MAP_ANONYMOUS , finalmente offset debe de ser un múltiplo de la memoria pero es cero por MAP_ANONYMOUS. Y devuelve el lugar donde está realmente asignado el proceso.

```
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
```

access revisa los privilegios reales del usuario para entra al directorio dado en modo de lectura, devolviendo un -1 como error por no existir el directorio.

```
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
```

Devuelve un descriptor de archivo para el directorio dado en modo de lectura y protección hacia otros procesos que hacen llamadas al sistema. El descriptor no puede ser negativo, debe ser pequeño y entero.

```
fstat(3, {st_mode=S_IFREG|0644, st_size=321107, ...}) = 0
```

Devuelve informacion de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
mmap(NULL, 321107, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb29000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso es null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura, posteriormente crea una copia para escritura, el 3 es por descriptor de archivo, finalmente offset debe de ser un múltiplo de la memoria. Y devuelve el lugar donde está realmente asignado el proceso.

```
close(3)                                = 0
```

Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
open("/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
```

Devuelve un descriptor de archivo para el directorio dado en modo de lectura y protección hacia otros procesos que hacen llamadas al sistema. El descriptor no puede ser negativo, debe ser pequeño y entero.

```
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360\10\2\0\0\0\0\0"..., 832) = 832
```

Lee los bytes del descriptor de archivo e incrementa el offset en esa cantidad.

```
fstat(3, {st_mode=S_IFREG|0755, st_size=2076848, ...}) = 0
```

Devuelve informacion de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
mmap(NULL, 3967456, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fabcc58c000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso es null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura y poder ejecutar contenido, posteriormente crea una copia para escritura, MAP_DENYWRITE se ignora, el 3 es por el descriptor de archivo, finalmente offset debe de ser un múltiplo de la memoria. Y devuelve el lugar donde está realmente asignado el proceso.

```
mprotect(0x7fabcc74c000, 2093056, PROT_NONE) = 0
```

Cambia las protecciones de acceso de la memoria mapeada a partir de la dirección dada, seguida de la longitud a proteger, después el tipo de modo indicando que no todos pueden acceder a ella y devolviendo un cero por no haber errores.

```
mmap(0x7fabcc94b000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bf000) = 0x7fabcc94b000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio (0x7fabcc94b000), después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura y escritura, posteriormente crea una copia para escritura, MAP_DENYWRITE se ignora y se pide colocar la dirección real de memoria mapeada, el 3 es por el descriptor de archivo, finalmente offset debe de ser un múltiplo de la memoria. Y devuelve el lugar donde está realmente asignado el proceso.

```
mmap(0x7fabcc951000, 14816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fabcc951000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio (0x7fabcc951000), después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura y escritura, posteriormente crea una copia para escritura, se pide colocar la dirección real de memoria mapeada, MAP_ANONYMOUS inicializa a cero el contenido, se emplea -1 en lugar del descriptor de archivo, finalmente offset es cero por MAP_ANONYMOUS. Y devuelve el lugar donde está realmente asignado el proceso.

```
close(3)                                = 0
```

Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fabccb28000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura y escritura, posteriormente crea una copia para escritura, MAP_ANONYMOUS inicializa a cero el contenido, se emplea -1 en lugar del descriptor de archivo, finalmente offset es cero por MAP_ANONYMOUS. Y devuelve el lugar donde está realmente asignado el proceso.

```
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fabccb27000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura y escritura, posteriormente crea una copia para escritura, MAP_ANONYMOUS inicializa a cero el contenido, se emplea -1 en lugar del descriptor de archivo, finalmente offset es cero por MAP_ANONYMOUS. Y devuelve el lugar donde está realmente asignado el proceso.

```
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fabccb26000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura y escritura, posteriormente crea una copia para escritura, MAP_ANONYMOUS inicializa a cero el contenido, se emplea -1 en lugar del descriptor de archivo, finalmente offset es cero por MAP_ANONYMOUS. Y devuelve el lugar donde está realmente asignado el proceso.

```
arch_prctl(ARCH_SET_FS, 0x7fabccb27700) = 0
```

Selecciona una arquitectura para el proceso en este caso 64 bits después se le pasa como argumento una dirección (0x7fabccb27700) y regresa cero por no haber error.

```
mprotect(0x7fabcc94b000, 16384, PROT_READ) = 0
```

Cambia las protecciones de acceso de la memoria mapeada a partir de la dirección dada (0x7fabcc94b00), seguida de la longitud a proteger, después el tipo de modo indicando que se puede leer y devolviendo un cero por no haber errores.

```
mprotect(0x60f000, 4096, PROT_READ)      = 0
```

Cambia las protecciones de acceso de la memoria mapeada a partir de la dirección dada (0x60f000), seguida de la longitud a proteger, después el tipo de modo indicando que se puede leer y devolviendo un cero por no haber errores.

```
mprotect(0x7fabccb79000, 4096, PROT_READ) = 0
```

Cambia las protecciones de acceso de la memoria mapeada a partir de la dirección dada (0x7fabcc79000), seguida de la longitud a proteger, después el tipo de modo indicando que se puede leer y devolviendo un cero por no haber errores.

```
munmap(0x7fabccb29000, 321107)           = 0
```

Elimina la asociación de la memoria mapeada desde la dirección de inicio por la longitud dada.

```
brk(NULL)                                = 0x1502000
```

brk aumenta la memoria del proceso, cuando se le da null solo devuelve la dirección final de la memoria del proceso.

```
brk(0x1523000)                           = 0x1523000
```

Aumenta la memoria del proceso en la dirección dada.

```
open("/usr/lib64/locale/locale-archive", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
```

Devuelve un descriptor de archivo para el directorio dado en modo de lectura y nuevo descriptor de archivo. Devuelve -1 por no poder abrir el archivo.

```
open("/usr/share/locale/locale.alias", O_RDONLY|O_CLOEXEC) = 3
```

Devuelve un descriptor de archivo para el directorio dado en modo de lectura y nuevo descriptor de archivo. Devuelve 3 como descriptor de archivo.

```
fstat(3, {st_mode=S_IFREG|0644, st_size=2997, ...}) = 0
```

Devuelve información de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
read(3, "# Locale name alias data base.\n#"..., 1024) = 1024
```

Lee los bytes del descriptor de archivo e incrementa el offset en esa cantidad.

```
read(3, "nd for the time being for\n# back"..., 1024) = 1024
```

Lee los bytes del descriptor de archivo e incrementa el offset en esa cantidad.

```
read(3, "59-1\ngalego\t\tgl_ES.ISO-8859-1\nga"..., 1024) = 949
```

Lee los bytes del descriptor de archivo e incrementa el offset en esa cantidad.

```
read(3, "", 1024)                       = 0
```

Lee los bytes del descriptor de archivo e incrementa el offset en esa cantidad.

```
close(3)                                = 0
```

Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
open("/usr/lib64/locale/es_ES.utf8/LC_IDENTIFICATION", O_RDONLY|O_CLOEXEC) = 3
```

Devuelve un descriptor de archivo para el directorio dado en modo de lectura y nuevo descriptor de archivo. Devuelve 3 como descriptor de archivo.

```
fstat(3, {st_mode=S_IFREG|0644, st_size=353, ...}) = 0
```

Devuelve información de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
mmap(NULL, 353, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb77000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura, posteriormente crea una copia para escritura, se pasa el 3 como descriptor de archivo, finalmente offset es cero. Y devuelve el lugar donde está realmente asignado el proceso.

```
close(3)                                = 0
```

Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
open("/usr/lib64/gconv/gconv-modules.cache", O_RDONLY) = -1 ENOENT (No such file
or directory)
```

Devuelve un descriptor de archivo para el directorio dado en modo de lectura. Devuelve -1 por no poder abrir el archivo.

```
open("/usr/lib64/gconv/gconv-modules", O_RDONLY|O_CLOEXEC) = 3
```

Devuelve un descriptor de archivo para el directorio dado en modo de lectura y nuevo descriptor de archivo. Devuelve 3 como descriptor de archivo.

```
fstat(3, {st_mode=S_IFREG|0644, st_size=56095, ...}) = 0
```

Devuelve información de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
read(3, "# GNU libc iconv configuration.\n"..., 1024) = 1024
```

Lee los bytes del descriptor de archivo e incrementa el offset en esa cantidad.

```
read(3, "ssion matching results.\n#  filen"..., 1024) = 1024
read(3, ".4-1985-2//\nalias\tCSA7-2//\t\tCSA_"..., 1024) = 1024
read(3, "GB1988//\t\tGB_1988-80//\nalias\tGB_"..., 1024) = 1024
read(3, "1002//\tJUS_I.B1.002//\nmodule\tJUS"..., 1024) = 1024
read(3, "as\tISO646-FR1//\t\tNF_Z_62-010_197"..., 1024) = 1024
read(3, "NTERNAL\t\tISO646\t\t2\nmodule\tINTERN"..., 1024) = 1024
read(3, "\t\tINTERNAL\t\tISO8859-1\t1\nmodule\tI"..., 1024) = 1024
read(3, "ISO-IR-110//\t\tISO-8859-4//\nalias"..., 1024) = 1024
read(3, "8859-6//\nalias\tISO88596//\t\tISO-8"..., 1024) = 1024
read(3, "8859-8//\nalias\tISO_8859-8:1988//"..., 1024) = 1024
read(3, "10//\nalias\tISO_8859-10:1992//\tIS"..., 1024) = 1024
read(3, "\t\tISO-8859-14//\nalias\tISO_8859-1"..., 1024) = 1024
read(3, "8BIT//\tT.61-8BIT//\nalias\tT.618BI"..., 1024) = 1024
read(3, "\tmodule\t\tcost\nalias\tKOI8//\t\t\tKOI"..., 1024) = 1024
read(3, "s\tCSEBCDICATDE//\t\tEBCDIC-AT-DE//"..., 1024) = 1024
read(3, "IC-ES//\nalias\tEBCDICES//\t\tEBCDIC"..., 1024) = 1024
read(3, "CDIC-FR\t1\nmodule\tINTERNAL\t\tEBCDI"..., 1024) = 1024
read(3, "A//\t\tIBM037//\nalias\tEBCDIC-CP-WT"..., 1024) = 1024
read(3, "\tCSIBM275//\t\tIBM275//\nmodule\tIBM"..., 1024) = 1024
read(3, "DIC-CP-ES//\t\tIBM284//\nalias\tCSIB"..., 1024) = 1024
read(3, "\t\t1\nmodule\tINTERNAL\t\tIBM420//\t\tI"..., 1024) = 1024
read(3, "//\nalias\t850//\t\t\tIBM850//\nalias\t"..., 1024) = 1024
read(3, "/\t\t\tIBM857//\nalias\tCSIBM857//\t\tI"..., 1024) = 1024
read(3, "//\nalias\tCSIBM864//\t\tIBM864//\nal"..., 1024) = 1024
read(3, "\nmodule\tIBM869//\t\tINTERNAL\t\tIBM8"..., 1024) = 1024
read(3, "lias\tOSF1002037B//\t\tIBM891//\nmod"..., 1024) = 1024
read(3, "AL\t\tIBM922\t\t1\nmodule\tINTERNAL\t\tI"..., 1024) = 1024
read(3, "BM939//\nmodule\tIBM939//\t\tINTERNA"..., 1024) = 1024
read(3, "\t\tto\t\t\tmodule\t\tcost\nalias\tIBM-11"..., 1024) = 1024
read(3, "\tCSIBM1133//\t\tIBM1133//\nmodule\tI"..., 1024) = 1024
```

```
read(3, "rom\t\t\tto\t\t\tmodule\t\tcost\nalias\tMS"..., 1024) = 1024
read(3, "EUC-CN//\nalias\tCN-GB//\t\t\tEUC-CN/"..., 1024) = 1024
read(3, "-1251//\t\tCP1251//\nmodule\tCP1251/"..., 1024) = 1024
read(3, "RNAL\t\tCP1257//\t\tCP1257\t\t1\n\n#\tfro"..., 1024) = 1024
read(3, "dule\tCP775//\t\t\tINTERNAL\t\tCP775\t\t"..., 1024) = 1024
read(3, "T//\nmodule\tISO-2022-CN-EXT//\tINT"..., 1024) = 1024
read(3, "X3.110//\t\tANSI_X3.110\t1\n\n#\tfrom\t"..., 1024) = 1024
read(3, "INTERNAL\t\tGOST_19768-74\t1\nmodule"..., 1024) = 1024
read(3, "NTERNAL\t\tINIS-8//\t\tINIS-8\t\t1\n\n#\t"..., 1024) = 1024
read(3, "//\t\tISO_5428//\nalias\tISO_5428:19"..., 1024) = 1024
read(3, "SDANO//\t\tNATS-DANO//\nmodule\tNATS"..., 1024) = 1024
read(3, "/\t\tISIRI-3342//\nmodule\tISIRI-334"..., 1024) = 1024
read(3, "2\t\t1\n\n#\tfrom\t\t\tto\t\t\tmodule\t\tcost"..., 1024) = 1024
read(3, "CII-8\t1\n\n#\tfrom\t\t\tto\t\t\tmodule\t\tc"..., 1024) = 1024
read(3, "t\nalias\tIBM-1122//\t\tIBM1122//\nal"..., 1024) = 1024
read(3, "156//\nmodule\tIBM1156//\t\tINTERNAL"..., 1024) = 1024
brk(0x1544000)                          = 0x1544000
read(3, "st\nalias\tIBM-921//\t\tIBM921//\nali"..., 1024) = 1024
read(3, "\tfrom\t\t\tto\t\t\tmodule\t\tcost\nalias\t"..., 1024) = 1024
read(3, "\nalias\tCSIBM1144//\t\tIBM1144//\nmo"..., 1024) = 1024
read(3, "ERNAL\t\tIBM1149//\t\tIBM1149\t\t1\n\n#\t"..., 1024) = 1024
read(3, "971//\nalias\tCP4971//\t\tIBM4971//\n"..., 1024) = 1024
read(3, "712//\t\tINTERNAL\t\tIBM12712\t\t1\nmod"..., 1024) = 1024
read(3, "\n\n#\tfrom\t\t\tto\t\t\tmodule\t\tcost\nali"..., 1024) = 1024
read(3, "module\tINTERNAL\t\tISO-8859-9E//\t\t"..., 1024) = 799
read(3, "", 1024)                       = 0
close(3)                                = 0
open("/usr/lib64/locale/es_ES.utf8/LC_MEASUREMENT", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=23, ...}) = 0
mmap(NULL, 23, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb76000
close(3)                                = 0
open("/usr/lib64/locale/es_ES.utf8/LC_TELEPHONE", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=49, ...}) = 0
mmap(NULL, 49, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb75000
close(3)                                = 0
open("/usr/lib64/locale/es_ES.utf8/LC_ADDRESS", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=143, ...}) = 0
mmap(NULL, 143, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb74000
```

```
close(3)                                = 0
open("/usr/lib64/locale/es_ES.utf8/LC_NAME", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=62, ...}) = 0
mmap(NULL, 62, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb73000
close(3)                                = 0
open("/usr/lib64/locale/es_ES.utf8/LC_PAPER", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=34, ...}) = 0
mmap(NULL, 34, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb72000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura, posteriormente crea una copia para escritura, se pasa el 3 como descriptor de archivo, finalmente offset es cero. Y devuelve el lugar donde está realmente asignado el proceso.

```
close(3)                                = 0
```

Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
open("/usr/lib64/locale/es_ES.utf8/LC_MESSAGES", O_RDONLY|O_CLOEXEC) = 3
```

Devuelve un descriptor de archivo para el directorio dado en modo de lectura y nuevo descriptor de archivo. Devuelve 3 como descriptor de archivo.

```
fstat(3, {st_mode=S_IFDIR|0755, st_size=38, ...}) = 0
```

Devuelve información de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
close(3)                                = 0
```

Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
open("/usr/lib64/locale/es_ES.utf8/LC_MESSAGES/SYS_LC_MESSAGES",
O_RDONLY|O_CLOEXEC) = 3
```

Devuelve un descriptor de archivo para el directorio dado en modo de lectura y nuevo descriptor de archivo. Devuelve 3 como descriptor de archivo.

```
fstat(3, {st_mode=S_IFREG|0644, st_size=58, ...}) = 0
```

Devuelve información de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
mmap(NULL, 58, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb71000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura, posteriormente crea una copia para escritura, se pasa el 3 como descriptor de archivo, finalmente offset es cero. Y devuelve el lugar donde está realmente asignado el proceso.

```
close(3)                                = 0
```
Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
open("/usr/lib64/locale/es_ES.utf8/LC_MONETARY", O_RDONLY|O_CLOEXEC) = 3
```
Devuelve un descriptor de archivo para el directorio dado en modo de lectura y nuevo descriptor de archivo. Devuelve 3 como descriptor de archivo.

```
fstat(3, {st_mode=S_IFREG|0644, st_size=294, ...}) = 0
```
Devuelve información de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
mmap(NULL, 294, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb70000
```
mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura, posteriormente crea una copia para escritura, se pasa el 3 como descriptor de archivo, finalmente offset es cero. Y devuelve el lugar donde está realmente asignado el proceso.

```
close(3)                                = 0
```
Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
open("/usr/lib64/locale/es_ES.utf8/LC_TIME", O_RDONLY|O_CLOEXEC) = 3
```
Devuelve un descriptor de archivo para el directorio dado en modo de lectura y nuevo descriptor de archivo. Devuelve 3 como descriptor de archivo.

```
fstat(3, {st_mode=S_IFREG|0644, st_size=2378, ...}) = 0
```
Devuelve información de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
mmap(NULL, 2378, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb6f000
```
mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura, posteriormente crea una copia para escritura, se pasa el 3 como descriptor de archivo, finalmente offset es cero. Y devuelve el lugar donde está realmente asignado el proceso.

```
close(3)                                = 0
```
Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
open("/usr/lib64/locale/es_ES.utf8/LC_NUMERIC", O_RDONLY|O_CLOEXEC) = 3
```
Devuelve un descriptor de archivo para el directorio dado en modo de lectura y nuevo descriptor de archivo. Devuelve 3 como descriptor de archivo.

```
fstat(3, {st_mode=S_IFREG|0644, st_size=54, ...}) = 0
```

Devuelve información de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
mmap(NULL, 54, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb6e000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura, posteriormente crea una copia para escritura, se pasa el 3 como descriptor de archivo, finalmente offset es cero. Y devuelve el lugar donde está realmente asignado el proceso.

```
close(3)                                = 0
```

Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
open("/usr/lib64/locale/es_ES.utf8/LC_CTYPE", O_RDONLY|O_CLOEXEC) = 3
```

Devuelve un descriptor de archivo para el directorio dado en modo de lectura y nuevo descriptor de archivo. Devuelve 3 como descriptor de archivo.

```
fstat(3, {st_mode=S_IFREG|0644, st_size=317692, ...}) = 0
```

Devuelve información de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
mmap(NULL, 317692, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccad8000
```

mmap asocia un rango de direcciones físicas a la memoria de un dispositivo, primero se le da una dirección de inicio en este caso null, después la longitud en bytes, seguido del tipo de protección de la memoria en modo lectura, posteriormente crea una copia para escritura, se pasa el 3 como descriptor de archivo, finalmente offset es cero. Y devuelve el lugar donde está realmente asignado el proceso.

```
close(3)                                = 0
```

Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
open("/etc/localtime", O_RDONLY|O_CLOEXEC) = 3
```

Devuelve un descriptor de archivo para el directorio dado en modo de lectura y nuevo descriptor de archivo. Devuelve 3 como descriptor de archivo.

```
fstat(3, {st_mode=S_IFREG|0644, st_size=2619, ...}) = 0
```

Devuelve información de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
fstat(3, {st_mode=S_IFREG|0644, st_size=2619, ...}) = 0
```

Devuelve información de un archivo pero usando el descriptor de archivo (numero 3), establece un modo normal, da el tamaño del archivo en bytes. Como no hubo error regresa 0.

```
read(3, "TZif2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\t\0\0\0\t\0\0\0\0"..., 4096)
= 2619
```

Lee los bytes del descriptor de archivo e incrementa el offset en esa cantidad.

```
lseek(3, -1656, SEEK_CUR)                       = 963
```

Desplaza el archivo con descriptor de archive 3 en dirección actual más un agregado (-1656).

```
read(3, "TZif2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\t\0\0\0\t\0\0\0\0"..., 4096)
= 1656
```

Lee los bytes del descriptor de archivo e incrementa el offset en esa cantidad.

```
close(3)                                        = 0
```

Cierra el descriptor de archive con número 3 y como no ocurrió error devuelve 0.

```
fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(136, 1), ...}) = 0
```

Devuelve información de un archivo pero usando el descriptor de archivo (numero 1), establece un modo normal, da el tamaño del archivo en bytes. Describe el dispositivo donde esta el archivo. Como no hubo error regresa 0.

```
write(1, "dom ago 27 01:21:17 CEST 2017\n", 30dom ago 27 01:21:17 CEST 2017
) = 30
```

Escribe n bytes desde el buffer asociado al descriptor de archivo (1), devuelve el número de bytes escritos.

```
close(1)                                        = 0
```

Cierra el descriptor de archive con número 1 y como no ocurrió error devuelve 0.

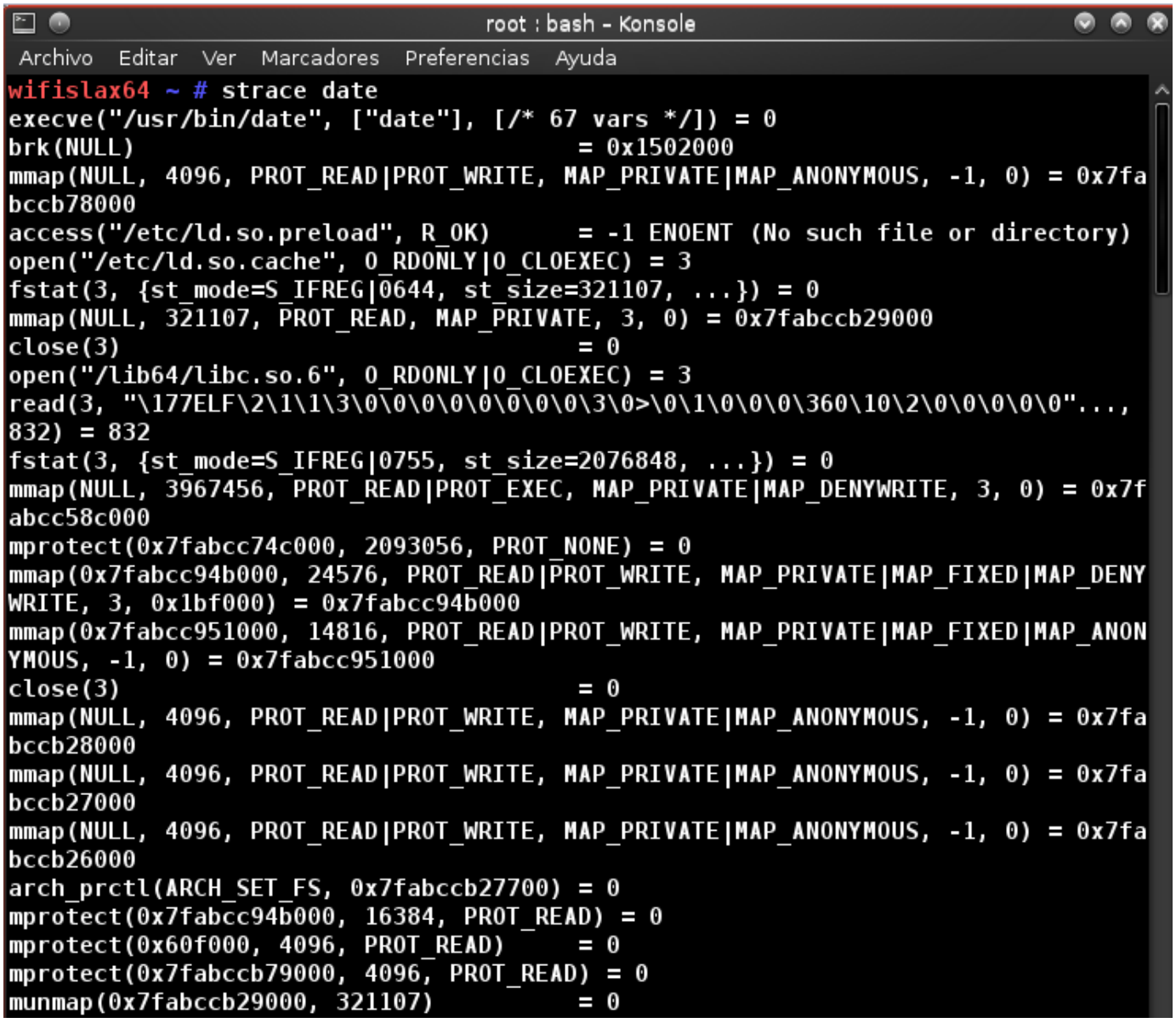
```
close(2)                                        = 0
```

Cierra el descriptor de archive con número 2 y como no ocurrió error devuelve 0.

```
exit_group(0)                                   = ?
```

Sale de todos los hilos en todos los procesos.

```
+++ exited with 0 +++
wifislax64 ~ #
```


root : bash - Konsole
Archivo Editar Ver Marcadores Preferencias Ayuda
wifislax64 ~ # strace date
execve("/usr/bin/date", ["date"], [/* 67 vars */]) = 0
brk(NULL) = 0x1502000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fa
bccb78000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=321107, ...}) = 0
mmap(NULL, 321107, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fabccb29000
close(3) = 0
open("/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360\10\2\0\0\0\0\0"...,
832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2076848, ...}) = 0
mmap(NULL, 3967456, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f
abcc58c000
mprotect(0x7fabcc74c000, 2093056, PROT_NONE) = 0
mmap(0x7fabcc94b000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENY
WRITE, 3, 0x1bf000) = 0x7fabcc94b000
mmap(0x7fabcc951000, 14816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANON
YMOUS, -1, 0) = 0x7fabcc951000
close(3) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fa
bccb28000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fa
bccb27000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fa
bccb26000
arch_prctl(ARCH_SET_FS, 0x7fabccb27700) = 0
mprotect(0x7fabcc94b000, 16384, PROT_READ) = 0
mprotect(0x60f000, 4096, PROT_READ) = 0
mprotect(0x7fabccb79000, 4096, PROT_READ) = 0
munmap(0x7fabccb29000, 321107) = 0