Strace i Ltrace

Strace:

La comanda *strace* serveix per conèixer quines "crides al sistema" (es a dir, crides a funcions proporcionades pel kernel com ara la d'obrir, llegir, escriure o tancar fitxers o sockets, per exemple) fa servir un determinat programa d'usuari. Per tant, t'eina *strace* ens pot ajudar a esbrinar què fa realment una determinada comanda: quins fitxers llegeix, quins vol escriure, quins ports obre, a quins ports es vol connectar, etc, etc. És com ser un "espia" de tot allò que el programa fa a baix nivell.

La sintaxi general de strace és: *strace comanda parametres*. on la comanda indicada s'executarà tal qual. Per exemple: *strace ls -l* executarà *ls -l* i, a mesura que avanci aquesta execució, anirà detectant i mostrant per pantalla (a stderr més en concret) totes les crides al sistema que realitzi *ls -l* La manera de mostrar-les és indicant, per cada crida detectada, primer el seu nom, després, entre parèntesis, els paràmetres concrets que aquesta crida ha rebut i, finalment, després d'un símbol "=", el valor retornat per aquesta crida (el significat del qual dependrà de la crida en qüestió: un valor igual a 0 sol significar "tot ok" i un valor diferent de 0 sol significar algun tipus d'error, encara que no sempre és així).

Com que les crides mostrades poden ser moltes depenent de la comanda inspeccionada, se sol afegir el paràmetre *-e crida1,crida2,..* per mostrar només les crides que ens interessen.Algunes crides comunes són:

open/openat : Obre un fitxer. Retorna un identificador numèric del fitxer anomenat "file descriptor" (fd)
access : Comprova els permisos d'un fitxer. Una crida més general seria "fstat" o també "statfs"
read : Llegeix un fitxer. Es pot filtrar exactament per quin fitxer si s'escriu read=nºfd
write : Escriu en un fitxer. Es pot filtrar exactament per quin fitxer si s'escriu write=nºfd
NOTA: El fd nº 1 representa stdout, el nº2 representa stderr i el nº0 representa stdin (el teclat)
lseek :Mou el cursor de memòria al llarg del fitxer (per poder llegir-hi o escriure-hi a la nova posició)
fchmod : Canvia els permisos d'un fitxer. També està "fchown" (que canvia el propietari)
close : Tanca un fitxer
unlink : Esborra un fitxer
chdir : Estableix el directori de treball del procés en qüestió
socket : Obre un socket local. Retorna també un "file descriptor" (fd) identificador del socket
connect : Obre una conexió des d'un socket local amb un socket remot (normalment per fer de client)
bind : Activa un socket local (pas previ imprescindible per posar-lo a l'escolta tot seguit amb "*listen*")
setsockopt : Defineix diferents opcions per un determinat socket. També està "accept".
sendto : Envia dades a un socket remot. També es pot usar "write"
recvfrom : Rep dades en un socket local. També es pot usar "read"
mmap/munmap : Carrega/descarrega dades a/de la memòria catxé de la RAM. També està "brk".
execve : Executa un programa. Sol ser la primera crida que apareix
system : Executa un shell script.
fork/clone : El programa es clona a sí mateix (normalment això es fa per convertir-se en dimoni)
kill : Envia una senyal a algun altre procés

NOTA: Per saber més sobre les crides al sistema disponibles recomano la lectura de *man syscalls*.

Altres valors que poden acompanyar al paràmetre -e i que són més genèrics són:

file : Representa qualsevol crida al sistema que tingui un fitxer com a paràmetre
process : Representa qualsevol crida al sistema relacionada amb la gestió de processos
memory : Representa qualsevol crida al sistema relacionada amb la gestió de memòria
network : Representa qualsevol crida al sistema relacionada amb la xarxa
signal : Representa qualsevol crida al sistema relacionada amb la gestió de senyals
ipc : Representa qualsevol crida al sistema relacionada amb mecanismes IPC
desc : Representa qualsevol crida al sistema relacionada amb descriptors de fitxers (read,write...)

Altres paràmetres de *strace* són:

-o nomFitxer : Guarda la sortida de *strace* en el fitxer indicat (en comptes de per stderr)
-C: Mostra estadístiques al final de la sortida habitual: nº de crides,temps consumit per crida, errors...
-c : Mostra només les estadístiques. Es pot combinar amb *-S {time|calls| name}* per ordenar la sortida
-p PID1, PID2 : Aplica *strace* a comandes que ja estiguin en marxa en temps real
-P /ruta/fitxer : Monitoritza (només) totes les crides que pateix /ruta/fitxer per un determinat procés
-t : Mostra també l'hora i dia en la que s'ha realitzat cada crida
-tt : Similar a l'anterior però incloent els microsegons
-ttt : Similar a l'anterior però mostrant el temps amb format "UNIX Epoch"
-r : Mostra també el temps relatiu entre crida i crida
-T : Mostra també el temps que triga cada crida (la diferència entre l'hora d'inici i la de final)
-s nº : quantitat de caràcters que se mostren/guarden per cada crida (per defecte 32)
-f : Mostra també les crides realitzades pels possibles processos fills que apareguin
-ff : En el cas d'usar -o nomFitxer, es guarden les crides de cada procés fill en un fitxer diferent (anomenat "nomFitxer.pid")

Ltrace:

Cal distingir entre "crides al sistema" i "crides a funcions estàndar". Les primeres, tal com hem dit, són crides a funcions C proporcionades directament pel kernel; les segones també són crides a funcions C però en aquest cas aquestes estan proporcionades per la llibreria estàndar utilitzada a l'hora de compilar el programa en qüestió, la qual sol ser la GNU Glibc (https://www.gnu.org/software/libc) . Cada funció de la llibreria estàndar sol actuar -encara que no sempre- com a "wrapper" (embolcalls) d'una (o d'un conjunt de) crida/es al sistema, perquè, en general, les primeres realitzen tasques de baix nivell d'una manera menys directa (i per tant, més controlada, segura i portable) que les segones, estalviant així al desenvolupador haver de fer ell mateix les crides al sistema directament (sempre més delicades i perilloses). Per exemple, l'execució de la funció *printf("Hola mundo");* ,proporcionada per la llibreria estàndar, acaba provocant una crida a una funció proporcionada pel kernel anomenada *write(1,"Hola mundo",10);* -on el nº1 representa el descriptor de la sortida estàndar, stdout, i el nº10 és la quantitat de caràcters a escriure-. En tot cas, ja sigui perquè un programa cridi directament a una funció proporcionada pel kernel o bé perquè executi una funció proporcionada per la llibreria estàndar que realitza pel seu compte una crida a a una funció proporcionada pel kernel, *strace* registra i mostra totes aquestes crides sense distinció.

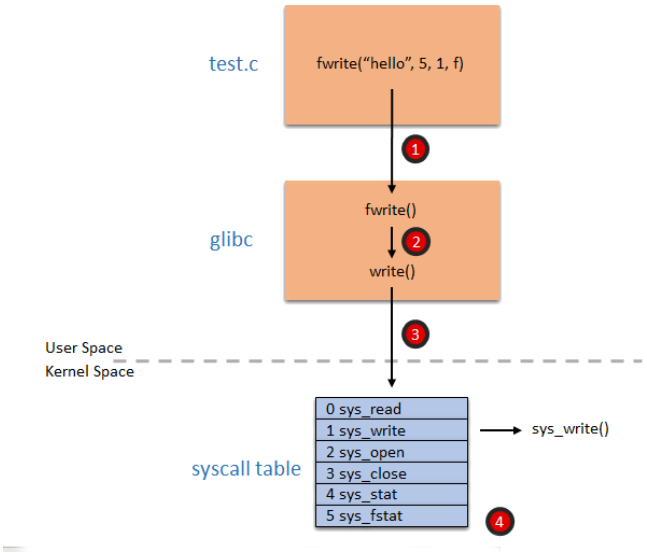
NOTA: No totes les funcions estàndar són"wrappers" de funcions del kernel. Per exemple, la funció *strlen()*, que serveix per conèixer la longitud d'una cadena passada com a paràmetre, no executa cap codi del kernel perquè no li cal.
NOTA: Per saber més sobre la llibreria estàndar, recomano la lectura de *man libc*

No obstant, si volem inspeccionar la crida a funcions incloses dins de (només) la llibreria estàndar, podem utilitzar una altra eina anomenada *ltrace* . Aquesta eina també admet el paràmetre -e, però en aquest cas, els valors que pot tenir són els noms de les funcions estàndars a inspeccionar (per exemple, *printf, malloc, free, strcmp, strlen, strcpy, getenv, opendir, readdir, closedir*, etc) i a més, concatenades mitjançant el símbol "+" en comptes de la coma. Per exemple, així: *ltrace -e opendir+readdir+closedir ls*. Altres paràmetres de *ltrace* són:

-o nomFitxer : Guarda la sortida de *strace* en el fitxer indicat (en comptes de per stderr)
-c : Mostra les estadístiques de quantes crides s'han fet a cada funció i quant de temps en total s'ha gastat per cadascuna
-p PID1 : Serveix per aplicar l*trace* a comandes que ja estiguin en marxa en temps real
-t : Mostra també l'hora i dia en la que s'ha realitzat cada crida
-tt : Similar a l'anterior però incloent els microsegons
-ttt : Similar a l'anterior però mostrant el temps amb format "UNIX Epoch"
-r : Mostra també el temps relatiu entre crida i crida
-T : Mostra també el temps que triga cada crida (la diferència entre l'hora d'inici i la de final)
-S : Mostra també les crides al sistema (marcades amb "SYS") dins les crides a la llibreria estàndar

Cal tenir en compte, no obstant, que aquesta comanda només funciona per inspecciona executables de tipus ELF (els nadius de Linux). Per saber si un determinat programa és ELF o no es pot utilitzar la comanda *file /ruta/programa*

A continuació es mostra un simple diagrama (extret de https://sysdig.com/blog/fascinating-world-linux-system-calls/) on es mostra gràficament el que s'acaba d'explicar:



Llibreries:

Per saber si un determinat programa està enllaçat o no a la llibreria estàndar i, en general, per saber a quines altres llibreries hi està enllaçat, es pot fer ***ldd /ruta/programa*** . Molt sovint la comanda *ldd* mostrarà alguna de les següents llibreries (perquè són molt utilitzades per multitud de programes) que cal conèixer:

*"/lib/libc.so.X" (on X és un número): la llibreria estàndar GNU LibC pròpiament dita.

*"/lib/linux-vdso.so" : conté al seu interior un conjunt sel.leccionat de crides al sistema que són "injectades" pel kernel automàticament a la memòria del procés que carrega aquesta llibreria. Aquestes crides sel.leccionades tenen la particularitat d'executar-se en el context d'usuari i no pas de kernel (trobareu més informació a http://man7.org/linux/man-pages/man7/vdso.7.html o https://en.wikipedia.org/wiki/VDSO) fent així que la seva crida sigui més ràpida en no haver de canviar de context per executar-se.

*"/lib/ld.so" o "/lib/ld-linux.so" : carregador de llibreries del sistema (l'anomenat "loader"). Aquesta llibreria és ella mateixa carregada sempre automàticament el kernel abans de què s'executi el propi codi del programa en qüestió. La seva funcionalitat és trobar ràpidament les llibreries (inclosa l'estàndar GNU) que el programa en qüestió necessita (és a dir, les llibreries a les quals està enllaçat) per procedir seguidament a la seva correcta execució. Per a què aquest "loader" pugui trobar les llibreries necessàries en cada cas utilitza un "mapa", que és el fitxer "/etc/ld.so.conf", el qual consisteix simplement en una llista de rutes de carpetes -una per línia- on haurà d'anar a buscar aquestes llibreries.

NOTA: No obstant, per a què aquest "mapa" sigui realment efectiu cal "compilar-lo" en el "mapa" que realment s'utilitza a la pràctica. Per fer aquesta compilació (necessària cada vegada que el contingut de "/etc/ls.so.conf" canvïi) cal executar la comanda ***ldconfig*** . El resultat de la "compilació" és un altre fitxer anomenat "/etc/ls.so.cache", el qual és un fitxer de text que conté les rutes exactes -una rera l'altra i incloent també el nom - de totes les llibreries que el "loader" reconeixerà.

NOTA: La comanda *ldconfig* se sol executar automàticament quan s'instal.la al sistema alguna llibreria nova. Com a paràmetre més interessant per executar-lo manualment té el paràmetre -p, el qual serveix per mostrar la llista de llibreries carregades en aquest moment concret d'entre totes les que hi hagi llistades a /etc/ld.so.cache.

NOTA: Existeix la possibilitat d'indicar rutes complementàries de carpetes on el "loader" buscarà llibreries sense haver de modificar l'arxiu "/etc/ld.so.conf" si s'indiquen (separades per ":") com a valor de la variable d'entorn LD_LIBRARY_PATH. Les rutes indicades allà tenen preferència sobre les indicades a "/etc/ld.so.conf", i això és utilitzat moltes vegades per "substituir" alguna llibreria del sistema per una llibreria pròpia equivalent (ubicada en un altre lloc). No obstant, el valor d'aquesta variable romandrà només fins que la màquina es reinicïi.

EXERCICIS:

1.-a) Esbrina *amb ldd* a quines llibreries està enllaçat el binari *date* i digues per a què serveixen cadascuna d'elles.

NOTA: Hauràs d'escriure la ruta absoluta del binari *date* per aconseguir-ho. Si no saps quina és, sempre pot fer abans *which date*

b) Observa amb *strace* totes les crides al sistema que realitza la comanda *date* i digues si apareix la crida *gettimeofday()* i/o la crida *time()* -o similars- per obtenir, respectivament, la data i hora actual.

NOTA: Si la resposta és no, la raó és que aquestes funcions es troben actualment implementades dins de la llibreria "linux-vdso" i això fa que romanguin invisibles per a *strace*.

c) ¿Quina és la funció de Lib C usada per la comanda *date* per obtenir l'hora UTC del sistema (de fet, la única que pot oferir-nos el kernel)? Pista: fes servir *ltrace*

d) Consulta la pàgina *man 5 localtime* per deduir per què, tal com es veu a la sortida de *strace*, la comanda *date* obre el fitxer /etc/localtime

e) Si sabem que amb el paràmetre -S de *ltrace* podem saber les crides al sistema que es realitzen dins de cada execució d'una funció estàndar, esbrina quina és la funció de Lib C usada per la comanda *date* que realitza l'apertura del fitxer /usr/lib/locale/locale-archive (pas necessari per tal d'obtenir la configuració regional actual -d'entre les possibles instal.lades- i poder així mostrar la data i hora del sistema en l'idioma adient)

NOTA: L'arxiu /usr/lib/locale/locale-archive conté al seu interior el conjunt "compilat" de configuracions regionals instal.lades al sistema i és el resultat de l'execució (en algun moment anterior) de la comanda *locale-gen* (a Ubuntu o Arch) o *localedef* (a Fedora). Per més informació sobre aquest arxiu, veieu https://wiki.archlinux.org/index.php/locale#Generating_locales o bé llegiu el següent paràgraf, extret de https://lists.centos.org/pipermail/centos/2012-September/129331.html :

"The file /usr/lib/locale/locale-archive is a memory-mapped file used by glibc (the gnu C library). This file contains the languages used over the system (for instance, man pages). This memory-mapping allows to read in the file as if it were in memory, avoiding system calls used to perform disk-read operations, therefore it allows much faster access. Memory-mapped files (as well as shared libraries) are counted as part of the virtual memory of processes (see "top" command, "VIRT" field). Therefore, the part of the locale-archive file mapped to memory adds up to the virtual memory of every processes that makes use of glibc (basically everything), while this is actually only once in memory. In other words, for every processes, the virtual memory overestimates the real memory of the process by, at least, the part of the locale-archive file which is memory-mapped."

2.-a) Utilitza el filtre adient per mostrar amb *strace* les crides al sistema relacionades amb la manipulació de fitxers realitzades per la comanda *ncat -l -p 5000* ¿I les relacionades amb la gestió de la xarxa?

aBIS) ¿Quines crides al sistema relacionades amb la gestió de xarxa es produeixen en executar la comanda *nc 127.0.0.1 5000*? ¿Quines són iguals i quines diferents respecte les usades en posar en marxa el servidor Netcat anterior?

b) Executa les següents comandes pas a pas i raona el resultat final que obtens:

touch a.txt
strace -c -o uno.log cp a.txt b.txt
strace -c -o dos.log cp ***-a*** *a.txt b.txt*
sed -E "s/ +/ /g" uno.log | cut -f6-7 -d" " | sort > unoNet.log
sed -E "s/ +/ /g" dos.log | cut -f6-7 -d" " | sort > dosNet.log
diff -y unoNet.log dosNet.log
rm a.txt uno dos**

c) ¿Per què és necessari escriure els símbols "2>&1" a la comanda *strace ls 2>&1 | grep "openat"*? Què fan?

d) ¿Què fa la comanda *strace -rp 1*?

3.-Crea un fitxer anomenat "hola.c" amb el següent contingut:

```
#include <stdio.h>
int main(int argc, char* argv[]) {
        printf("Hola SO!\n");
        return 0;
}
```

Tot seguit, compila'l executant la comanda *gcc -o hola.exe hola.c* . A partir d'aquí, respon:

a) ¿Quines llibreries obre l'executable "hola.exe" durant la seva execució? Compara-ho amb la sortida de la comanda *ldd*

b) ¿Quina crida a la llibreria estàndar realitza hola.exe?

4.-Després de llegir el següent article (https://www.linuxito.com/gnu-linux/nivel-basico/1083-librerias-compartidas-variables-de-entorno-y-permisos-en-linux), respon les preguntes:

a) ¿Quina diferència hi ha entre una llibreria estàtica i una dinàmica?

b) ¿Què és el "soname" d'una llibreria?