112673790 (Yatna Verma)

Ques 1

Following are observations on a large directory-

System calls made by `ls -l`

```
access, lgetxattr, statfs64, getxattr, connect, open, brk, close,
execve, fstat64,futex, getdents64, ioctl,_llseek,lstat64, mmap2,
mprotect,munmap, read, readlink, rt_sigaction, rt_sigprocmask,
set_robust_list, set_thread_area, set_tid_address, socket, stat64,
ugetrlimit, uname, write
```

System calls made by `ls`

```
access, statfs64, open, brk, close, execve, fstat64, getdents64,
mmap2, mprotect,munmap, read, rt_sigaction, rt_sigprocmask,
set_robust_list, set_thread_area, set_tid_address, stat64,
ugetrlimit, uname, write
```

Relevant differences -

System call	No. of Calls (for ls)	No. of Calls (for ls -l)
`stat64/ lstat64/ fstat64`	1 / 0 / 10	231 / 233 / 27
`open`	9	30
`read/ readlink`	7/ 0	17/ 6
`write`	39	233

1. -l option does more write calls (it need to write more bytes on the output)
2. -l option does more stats (because it needs a lot of information like permissions, size, time created, owner, group etc.)
3. Similarly it needs to read this data as well hence there are more read operations as well.

Ques 2

Ltrace calls for `ls` (which look like system calls)

```
closedir, opendir, readdir64, fclose, fflush, ioctl, isatty
```

Other utility like calls are -

```
 __ctype_get_mb_cur_max, __cxa_atexit, __errno_location, __fpending,
__freading __overflow, _setjmp, bindtextdomain, fileno,
fwrite_unlocked, memcpy, setlocale,realloc, malloc, strlen etc.
```

Observations -

1. Ltrace intercepts dynamic library calls as opposed to system calls (without -S flag)
2. Dynamic library functions may internally make system calls
   E.g
   readdir64() called during ls is a wrapper around `SYS_gendents64` system call.
   fclose() called during ls uses the `SYS_close` system call internally.
   malloc() uses `SYS_brk` to get more space.
3. A program may make system calls directly without using any dynamic library calls.
   E.g ls seems to make system calls like `mmap2` and `access` directly, which ltrace cannot intercept without special flags.

Ques.3

Used -f flag in strace to get child process info as well.
Also nano and firefox were opened without supplying any file in argument (i.e `strace -cf libreoffice`, `strace -cf firefox`)
Row 1 tells total open calls made by programs and the number which resulted in error.
Row 2 tells the total successful open calls made i.e (#total - #errors)

Operations	ls	nano	Openoffice (libreoffice)	firefox
open	9(0)	59(6)	1880(1120)	3868(1066)
successful open calls	9	53	760	2802

Ques.4

mmap() creates a new mapping of files in the virtual address space of the calling process. Thus it can be considered as loading a file. No mmap call gives error. Hence all calls are successful.

Operations	ls	nano	Openoffice (libreoffice)	firefox
mmap	19(0)	16(0)	848(0)	2302(0)
successful mmap calls	19	16	848	2302