1.分析以下代码中使用的linux系统调用。要求使用strace工具（参见<http://linuxtools-rst.readthedocs.org/zh_CN/latest/tool/strace.html>）

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

char cmd[256];

void main(){

int ret;

printf("Hello world, this is Linux!");

while(1){

printf(">");

fgets(cmd,256,stdin);

cmd[strlen(cmd)] = 0;

if(fork() == 0){

execlp(cmd,NULL);

perror(cmd);

exit(errno);

} else {

wait(&ret);

printf("child process return %d \n",ret);

}

}

}

使用strace工具运行，得到

execve("./test", ["./test"], [/\* 36 vars \*/]) = 0

brk(0) = 0x8556000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

mmap2(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xb7727000

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

open("/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat64(3, {st\_mode=S\_IFREG|0644, st\_size=62136, ...}) = 0

mmap2(NULL, 62136, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0xb7717000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

open("/lib/i386-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0@\226\1\0004\0\0\0"..., 512) = 512

fstat64(3, {st\_mode=S\_IFREG|0755, st\_size=1734120, ...}) = 0

mmap2(NULL, 1747676, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0xb756c000

mprotect(0xb7710000, 4096, PROT\_NONE) = 0

mmap2(0xb7711000, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a4) = 0xb7711000

mmap2(0xb7714000, 10972, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0xb7714000

close(3) = 0

mmap2(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xb756b000

set\_thread\_area({entry\_number:-1 -> 6, base\_addr:0xb756b900, limit:1048575, seg\_32bit:1, contents:0, read\_exec\_only:0, limit\_in\_pages:1, seg\_not\_present:0, useable:1}) = 0

mprotect(0xb7711000, 8192, PROT\_READ) = 0

mprotect(0x8049000, 4096, PROT\_READ) = 0

mprotect(0xb774a000, 4096, PROT\_READ) = 0

munmap(0xb7717000, 62136) = 0

fstat64(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0

mmap2(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xb7726000

write(1, "Hello world, this is Linux!\n", 28) = 28

fstat64(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0

mmap2(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xb7725000

write(1, ">", 1) = 1

read(0, "Orangeloo\n", 1024) = 10

clone(child\_stack=0, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0xb756b968) = 2779

wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 2}], 0, NULL) = 2779

--- SIGCHLD (Child exited) @ 0 (0) ---

write(1, "child process return 512 \n", 26) = 26

write(1, ">", 1) = 1

read(0, "exit\n", 1024) = 5

clone(child\_stack=0, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0xb756b968) = 2780

wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 2}], 0, NULL) = 2780

--- SIGCHLD (Child exited) @ 0 (0) ---

write(1, "child process return 512 \n", 26) = 26

write(1, ">", 1) = 1

read(0, 0xb7725000, 1024) = ? ERESTARTSYS (To be restarted)

--- SIGHUP (Hangup) @ 0 (0) ---

+++ killed by SIGHUP +++

根据老师提供的http://linuxtools-rst.readthedocs.org/zh\_CN/latest/tool/strace.html

设置返回值的输出位置.默认 为40.

-e expr

指定一个表达式,用来控制如何跟踪.格式如下:

[qualifier=][!]value1[,value2]...

qualifier只能是 trace,abbrev,verbose,raw,signal,read,write其中之一.value是用来限定的符号或数字.默认的 qualifier是 trace.感叹号是否定符号.例如:

-eopen等价于 -e trace=open,表示只跟踪open调用.而-etrace!=open表示跟踪除了open以外的其他调用.有两个特殊的符号 all 和 none.

注意有些shell使用!来执行历史记录里的命令,所以要使用\\.

-e trace=set

只跟踪指定的系统 调用.例如:-e trace=open,close,rean,write表示只跟踪这四个系统调用.默认的为set=all.

-e trace=file

只跟踪有关文件操作的系统调用.

-e trace=process

只跟踪有关进程控制的系统调用.

-e trace=network

跟踪与网络有关的所有系统调用.

-e strace=signal

跟踪所有与系统信号有关的 系统调用

-e trace=ipc

跟踪所有与进程通讯有关的系统调用

-e abbrev=set

设定 strace输出的系统调用的结果集.-v 等与 abbrev=none.默认为abbrev=all.

-e raw=set

将指 定的系统调用的参数以十六进制显示.

-e signal=set

指定跟踪的系统信号.默认为all.如 signal=!SIGIO(或者signal=!io),表示不跟踪SIGIO信号.

-e read=set

输出从指定文件中读出 的数据.例如:

-e read=3,5

-e write=set

输出写入到指定文件中的数据.

-o filename

将strace的输出写入文件filename

-p pid

跟踪指定的进程pid.

-s strsize

指定输出的字符串的最大长度.默认为32.文件名一直全部输出.

-u username

以username 的UID和GID执行被跟踪的命令

我还是看不出来用了什么调用。