Московский авиационный институт
(национальный исследовательский университет)
Институт № 8 «Информационные технологии и прикладная математика»

Лабораторная работа №1
по курсу «Операционные системы»

Студент: Минеева Светлана Алексеевна

Группа: М8О-210Б-21

Преподаватель: Миронов Е.С

Вариант №14
Оценка: _____
Дата: 26.12.2022
Подпись: _____

Москва, 2022

Содержание:

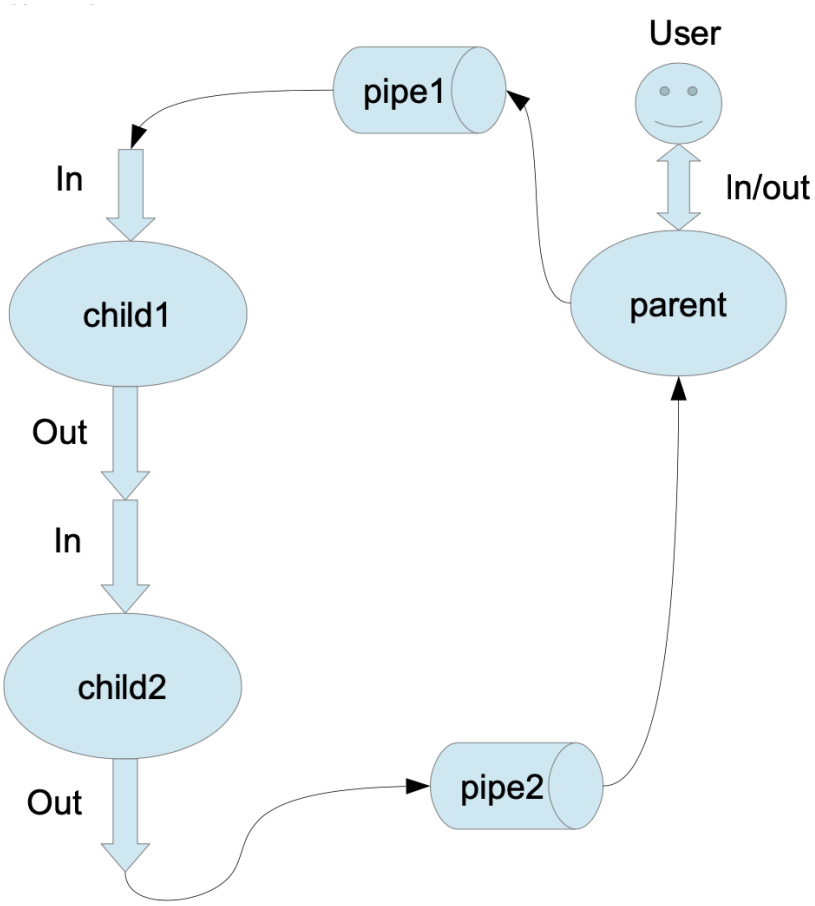
1. Цель работы

Приобретение практических навыков диагностики программного обеспечения.

2. Задание

Необходимо продемонстрировать ключевые системные вызовы, которые используются в лабораторной работе №2. Для этого будет использоваться утилита strace.

3. Вариант задания



Родительский процесс создает два дочерних процесса. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Child1 и Child2 можно «соединить» между собой дополнительным каналом. Родительский и дочерний процесс должны быть представлены разными программами.

Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child1 и child2 производят работу

над строками. Child2 пересылает результат своей работы родительскому процессу. Родительский процесс полученный результат выводит в стандартный поток вывода.

Вариант №14: Child1 переводит строки в нижний регистр. Child2 убирает все задвоенные пробелы.

4. Описание используемых утилит

Strace – это утилита Linux, отслеживающая системные вызовы, которые представляют собой механизм трансляции, обеспечивающий интерфейс между процессором и операционной системой. Использование данной утилиты позволяет понять, что процесс пытается сделать в данное время. Strace может быть очень полезен при отладке программ.

Для удобства работы с протоколом утилиты можно использовать следующие ключи:

- -o file – перенаправление протокола утилиты в файл file
- -p file – отслеживание исключительно обращений к файлу file
- -k – отображение стека вызова
- -f – отслеживание системных вызовов в дочерних процессах
- -y – замена в протоколе всех файловых дескрипторов на имена соответствующих им файлов (по возможности)

5. Протокол утилиты strace

Ниже приведен протокол работы утилиты strace, интересные места выделены с помощью других цветов.

```
elza@elza-NBLB-WAX9N:~$ strace ./parent
execve("./parent", ["./parent"], 0x7ffce7fd8520 /* 48 vars */) = 0
brk(NULL)                               = 0x56153e31c000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffe480dec0) = -1 EINVAL (Недопустимый аргумент)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=96553, ...}) = 0
mmap(NULL, 96553, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f5a8d6b0000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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\360q\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
```

```
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"...,
68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f5a8d6ae000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"...,
68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f5a8d4bc000
mprotect(0x7f5a8d4e1000, 1847296, PROT_NONE) = 0
mmap(0x7f5a8d4e1000, 1540096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f5a8d4e1000
mmap(0x7f5a8d659000, 303104, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0x7f5a8d659000
mmap(0x7f5a8d6a4000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f5a8d6a4000
mmap(0x7f5a8d6aa000, 13528, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f5a8d6aa000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f5a8d6af540) = 0
mprotect(0x7f5a8d6a4000, 12288, PROT_READ) = 0
mprotect(0x56153c666000, 4096, PROT_READ) = 0
mprotect(0x7f5a8d6f5000, 4096, PROT_READ) = 0
munmap(0x7f5a8d6b0000, 96553)           = 0
pipe([3, 4])                            = 0
pipe([5, 6])                            = 0
pipe([7, 8])                            = 0
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f5a8d6af810) = 5130
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f5a8d6af810) = 5131
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
brk(NULL)                               = 0x56153e31c000
brk(0x56153e33d000)                     = 0x56153e33d000
write(1, "Enter string:\n", 14Enter string:
)        = 14
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
read(0,       LLLL klk;o&HH
"      LLLL klk;o&HH\n", 1024)  = 20
write(4, " ", 1)                        = 1
read(7, " ", 1)                         = 1
write(1, " ", 1 )                       = 1
write(4, " ", 1)                        = 1
read(7, "\0", 1)                        = 1
write(1, "\0", 1)                       = 1
write(4, " ", 1)                        = 1
```

```
read(7, "\0", 1)                = 1
write(1, "\0", 1)                = 1
write(4, " ", 1)                 = 1
read(7, "\0", 1)                = 1
write(1, "\0", 1)                = 1
write(4, " ", 1)                 = 1
read(7, "\0", 1)                = 1
write(1, "\0", 1)                = 1
write(4, " ", 1)                 = 1
read(7, "\0", 1)                = 1
write(1, "\0", 1)                = 1
write(4, "L", 1)                 = 1
read(7, "l", 1)                 = 1
write(1, "l", 1l)                = 1
write(4, "L", 1)                 = 1
read(7, "l", 1)                 = 1
write(1, "l", 1l)                = 1
write(4, "L", 1)                 = 1
read(7, "l", 1)                 = 1
write(1, "l", 1l)                = 1
write(4, "L", 1)                 = 1
read(7, "l", 1)                 = 1
write(1, "l", 1l)                = 1
write(4, " ", 1)                 = 1
read(7, " ", 1)                 = 1
write(1, " ", 1 )                = 1
write(4, "k", 1)                 = 1
read(7, "k", 1)                 = 1
write(1, "k", 1k)                  = 1
write(4, "l", 1)                 = 1
read(7, "l", 1)                 = 1
write(1, "l", 1l)                = 1
write(4, "k", 1)                 = 1
read(7, "k", 1)                 = 1
write(1, "k", 1k)                  = 1
write(4, ";", 1)                 = 1
read(7, ";", 1)                 = 1
write(1, ";", 1;)                = 1
write(4, "o", 1)                 = 1
read(7, "o", 1)                 = 1
write(1, "o", 1o)                  = 1
write(4, "&", 1)                  = 1
read(7, "&", 1)                  = 1
write(1, "&", 1&)                    = 1
write(4, "H", 1)                  = 1
read(7, "h", 1)                 = 1
write(1, "h", 1h)                  = 1
write(4, "H", 1)                  = 1
read(7, "h", 1)                 = 1
write(1, "h", 1h)                  = 1
write(4, "\n", 1)                 = 1
read(7, "\n", 1)                = 1
```

```
write(1, "\n", 1
)                       = 1
read(0, 555 666
"555 666\n", 1024)              = 8
write(4, "5", 1)                  = 1
read(7, "5", 1)                   = 1
write(1, "5", 15)                   = 1
write(4, "5", 1)                  = 1
read(7, "5", 1)                   = 1
write(1, "5", 15)                   = 1
write(4, "5", 1)                  = 1
read(7, "5", 1)                   = 1
write(1, "5", 15)                   = 1
write(4, " ", 1)                  = 1
read(7, " ", 1)                   = 1
write(1, " ", 1 )                 = 1
write(4, "6", 1)                  = 1
read(7, "6", 1)                   = 1
write(1, "6", 16)                   = 1
write(4, "6", 1)                  = 1
read(7, "6", 1)                   = 1
write(1, "6", 16)                   = 1
write(4, "6", 1)                  = 1
read(7, "6", 1)                   = 1
write(1, "6", 16)                   = 1
write(4, "\n", 1)                 = 1
read(7, "\n", 1)                  = 1
write(1, "\n", 1
)                       = 1
read(0, "", 1024)                   = 0
exit_group(0)                     = ?
+++ exited with 0 +++
elza@elza-NBLB-WAX9N:~$
```

6. Описание системных вызовов

1) *execve("./parent", ["./parent"], 0x7ffce7fd8520 /* 48 vars */) = 0*

Исполняет программу ./parent с ключом ./parent и также передаются 48 переменных окружения.

2) *brk(NULL)   = 0x56153e31c000*

Устанавливает конец сегмента данных в значение NULL, возвращает указатель на начало новой области памяти.

3) *access("/etc/ld.so.preload", R_OK)   = -1 ENOENT (Нет такого файла или каталога)*

Проверяет /etc/ld.so.preload на существование и на наличие прав на чтение (R_OK), возвращает -1, если не существует или нет прав на чтение.

4) *openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3*

Открывает /etc/ld.so.cache относительно каталога вызывающего процесса (AT_FDCWD) с правами доступа на чтение и закрытие при завершении процесс (O_RDONLY|O_CLOEXEC). Возвращает файловый дескриптор для файла.

5) *fstat(3, {st_mode=S_IFREG|0644, st_size=96553, ...}) = 0*

Заполняет структуру, указанную вторым аргументом, fstat информацией о файле с файловым дескриптором 3.

6) *mmap(NULL, 96553, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f5a8d6b0000*

Создает отображение файла с файловым дескриптором 3 в память, начиная с адреса NULL, то есть ядро система само определит адрес, длины 96553 байт, с правами на чтение (PROT_READ), создает неразделяемое отражение с механизмом copy-on-write (MAP_PRIVATE), со смещением в файловом дескрипторе равным 0. Возвращает указатель на начало отраженной памяти = 0x7f5a8d6b0000.

7) *close(3) = 0*

Закрывает файловый дескриптор.

8) *read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"..., 832) = 832*

Читает 832 байта из 3 файлового дескриптора. Возвращает количество прочитанных байт.

9) *pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784*

Читает 784 байт из 3 файлового дескриптора с начальным смещением равным 64. Смещение для файлового дескриптора не изменяется. Возвращает количество прочитанных байт.

10) *mprotect(0x7f5a8d4e1000, 1847296, PROT_NONE) = 0*

Контролирует доступ к области памяти, начинающейся с адреса 0x7f5a8d4e1000 и длиной 1847296 байт, доступ к памяти запрещен (PROT_NONE). Если программой производится запрещенный этой функцией доступ к памяти, то такая программа получает сигнал SIGSEGV.

11) *arch_prctl(ARCH_SET_FS, 0x7f5a8d6af540) = 0*

Устанавливает специфичное для архитектуры состояние процесса или треда. Устанавливает 64 битную базу для регистра FS (ARCH_SET_FS) в значение 0x7f5a8d6af540.

12) *munmap(0x7f5a8d6b0000, 96553) = 0*

Удаляет все отражения, начиная с адреса 0x7f5a8d6b0000 длины 96553.

13) *pipe([5, 6]) = 0*

Создает пару файловых описателей, указывающих на запись inode именованного канала, и помещает их в массив. Файловый описатель, равный 5, предназначен для чтения, а 6 - для записи.

14) *clone(child_stack=NULL,*
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f5a8d6af810) = 5130

Создаёт новый процесс. Очищает TID для ребенка, но не для родителя, записывает TID ребенка в адрес 0x7f5a8d6af810. Создает сигнал для родителя SIGCHLD, вызываем при изменении статуса ребенка. Возвращает TID ребенка.

15) *write(4, "k", 1) = 1*

Записывает в файловый дескриптор 4 строку k размером 1 байт. Возвращает количество записанных байт.

7. Вывод

В результате выполнения данной лабораторной работы я приобрела важные практические навыки диагностики программного обеспечения. Я поняла, что утилита strace – хороший инструмент для отслеживания системных вызовов. Она очень полезна при отладке и тестировании программ, что пригодится мне в будущем. Протокол этой утилиты сначала кажется громоздкой и запутанной, но на деле, разобравшись в системных вызовах, он оказывается весьма хорошо читаемым, причем протокол утилиты можно заранее отредактировать с помощью различных ключей и подать на вывод только интересующую информацию.