Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №8 по курсу
«Операционные системы»

Тема работы
“Утилита strace”

Студент: Ковриженков Дмитрий Олегович
Группа: М8О-203Б-23

Преподаватель: Миронов Евгений Сергеевич

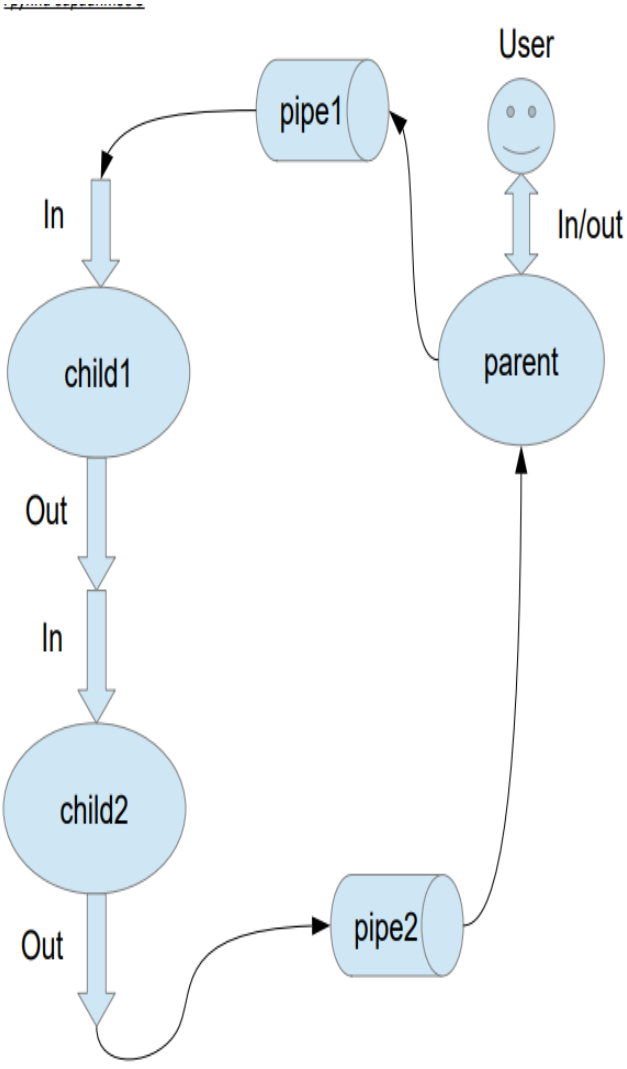
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Лабораторная работа №1

Задача: 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.

В 13 варианте Child1 переводит строки в нижний регистр. Child2 превращает все пробельные символы в символ «_».



Strace по лабораторной работе представлен в приложении 1.

Сначала программа запускается системным вызовом execve, который загружает и начинает выполнение исполняемого файла ./lab3. Затем происходит выделение памяти с использованием системных вызовов brk и mmap, что необходимо для корректной работы программы.

Программа проверяет системные файлы, такие как /etc/ld.so.preload, с помощью вызова access, а затем открывает и загружает динамические библиотеки, включая libc, с использованием системных вызовов openat, read и mmap. После загрузки библиотек создается процесс с использованием вызова fork(), а взаимодействие между процессами осуществляется через общую память с помощью mmap().

После запуска программа ожидает ввод строки от пользователя с помощью read(). Затем создается общий сегмент памяти с mmap(), и введенная строка записывается в него. Дочерние процессы child1 и child2 получают доступ к этому сегменту, выполняют необходимые преобразования и изменяют данные. child1 переводит строку в нижний регистр, а child2 заменяет пробелы на _. Затем родительский процесс считывает данные из общей памяти и выводит их с помощью write().

После завершения работы память освобождается через munmap(), а процессы корректно завершаются с помощью exit().

Приложение 1.

```
execve("./lab3", ["./lab3"], 0x7fff6c83fef0 /* 76 vars */) = 0

brk(NULL)                               = 0x5873da86d000

arch_prctl(0x3001 /* ARCH_??? */, 0x7fff752a7d00) = -1 EINVAL
(Недопустимый аргумент)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7d50c2d000

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

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=91319, ...},
AT_EMPTY_PATH) = 0

mmap(NULL, 91319, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f7d50c16000

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\0P\237\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
\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236
S"..., 68, 896) = 68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...},
AT_EMPTY_PATH) = 0

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

mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f7d50800000

mprotect(0x7f7d50828000, 2023424, PROT_NONE) = 0

mmap(0x7f7d50828000, 1658880, PROT_READ|PROT_EXEC, MAP_PRI-
VATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f7d50828000

mmap(0x7f7d509bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENY-
WRITE, 3, 0x1bd000) = 0x7f7d509bd000

mmap(0x7f7d50a16000, 24576, PROT_READ|PROT_WRITE, MAP_PRI-
VATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7f7d50a16000

mmap(0x7f7d50a1c000, 52816, PROT_READ|PROT_WRITE, MAP_PRI-
VATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f7d50a1c000

close(3)                                = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x7f7d50c13000

arch_prctl(ARCH_SET_FS, 0x7f7d50c13740) = 0

set_tid_address(0x7f7d50c13a10)         = 20414

set_robust_list(0x7f7d50c13a20, 24)     = 0

rseq(0x7f7d50c140e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f7d50a16000, 16384, PROT_READ) = 0

mprotect(0x5873d8be8000, 4096, PROT_READ) = 0

mprotect(0x7f7d50c67000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_IN-
FINITY}) = 0

munmap(0x7f7d50c16000, 91319)           = 0
```

```
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SET-
TID|SIGCHLD, child_tidptr=0x7f7d50c13a10) = 20415

wait4(-1, NULL, 0, NULL)                = ? ERESTARTSYS (To be restarted
if SA_RESTART is set)

--- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---

+++ killed by SIGINT +++
```