Лабораторная работа № 1
по курсу "Операционные системы":

Выполнил студент группы 08-208 МАИ *Куликов Алексей*.

Цель работы

Приобретение практических навыков диагностики работы программного обеспечения.

Задание

При выполнении последующих лабораторных работ необходимо продемонстрировать ключевые системные вызовы, которые в них используются и то, что их использование соответствует варианту ЛР. По итогам выполнения всех лабораторных работ отчет по данной должен содержать краткую сводку по исследованию последующих ЛР.

Ввиду того, что все работы по данному курсу выполнялись, выполняются и будут выполняться на операционной системе семейства UNIX, в качестве конекретного средства диагностики будет использована утилита `strace`.

Описание

`strace` – утилита, предназначенная для отслеживания системных вызовов, которые совершает процесс в ходе работы и сигналов, получаемых им.

Демонстрация работы

В качестве примера можно посмотреть что происходит при вызове так часто используемой всеми пользователями UNIX подобных систем утилиты `ls`.

На моей системе результат работы `strace` выглядит как-то так:

```
strace ls
execve("/bin/ls", ["ls"], 0x7ffc0c43c520 /* 54 vars */) = 0
brk(NULL)                               = 0x5622604ff000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=145421, ...}) = 0
mmap(NULL, 145421, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff8cc782000

...
close(3)                                = 0
fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(136, 2), ...}) = 0
write(1, "trace.txt\n", 10)             = 10
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Как можно видеть, даже такая, с первого взгляда, простая программа очень активно использует системные вызовы в своих целях (всего там было 94 строчки системных вызовов).

Что все это означает придется выяснить в ходе изучения данного курса, пользуясь этой утилитой.

Выполнения лабораторной

Во время работы над лабораторной работой №2 предстоит выяснить как работать с процессами, как организовать их взаимодействие и какие системные вызовы при этом используются.

В процессе выполнения лабораторной №3 будет изучены основы мультипоточного программирвания, и взаимодействия потоков с помощью примитивов синхронизации. И, соответственно, что у них "под капотом"тоже будет выяснено при помощи `strace`.

При выполнении 4-й лабораторной будем ознакомлены с разделяемой памятью процесса и файловыми отображениями. Все это отразится в выводе утилиты.

В 5-й лабораторной работе произойдет знакомство с динамическими библиотеками. Тут `strace` покажет как происходит поиск библиотеки при загрузке ее в память, отображение машинного кода в память процесса и взаимодействие с ним.

В конце, в лабораторной №6 будет изучены и опробованы на практике основы работы с очередями сообщений, предназначенных для межпроцессного взаимодействия (и не только). Как они устроены внутри (в общих чертах), опять же, покажет утилита для диагностики.

UPD: все вышесказанное было проделано, материал изучен. С помощью данной утилиты были проведены наблюдения за работой созданных программ. Было выяснено какие вызовы за что отвечают (основные) и отражено в отчетах по соответствующим лабораторным работам в секции «Strace».

Выводы

`strace` – удобная утилита для наблюдения за процесами. Знать, как программа взаимодействует с операционной системой, зачастую, оказывается довольно полезным. Это может потребоваться как при отладке программы на этапе разработки, так и при поиске причин сбоев работы чужой программы программы. А потом баг-репорты, благодарность от разработчиков и слава. Так же ее можно использовать в образовательных целях (для чего она, собственно, нам и понадобилась) на этапе изучения программирования в UNIX системах.

С помощью данной утилиты во время выполнения лабораторных работы были получено понимание того, какие системные вызовы скрываются под вызовом той или иной функции или в работе некоторых библиотек. Полезность данной утилиты сложно переоценить.