

Лабораторная работа №14

Операционные системы

Павлова Татьяна Юрьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	9
5	Выводы	14

Список иллюстраций

4.1	lab14-1.sh	10
4.2	Компиляция файла	10
4.3	ls /usr/share/man/man1	11
4.4	lab14-2.sh	11
4.5	реализация команды man kill	12
4.6	Компиляция файла	12
4.7	lab14-3.sh	12
4.8	Компиляция файла	13

Список таблиц

1 Цель работы

Целью данной лабораторной работы, является изучение основ программирования в оболочке ОС UNIX, а также научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2 Задание

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (`>/dev/tty#`, где `#` — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов.
2. Реализовать команду `man` с помощью командного файла. Изучите содержимое каталога `/usr/share/man/man1`. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой `less` сразу же просмотрев содержимое справки. Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1`.
3. Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что `$RANDOM` выдаёт псевдослучайные числа в диапазоне от 0 до

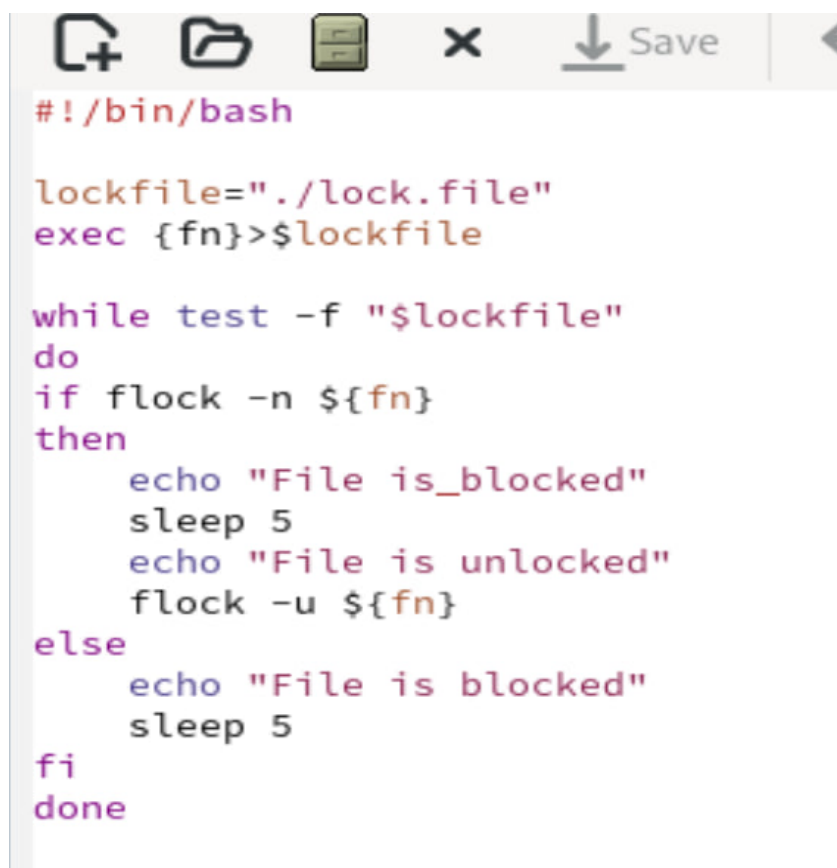
32767.

3 Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: — оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; — C-оболочка (или csh) — надстройка на оболочке Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; — оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; — BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна.

4 Выполнение лабораторной работы

Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (`>/dev/tty#`, где #—номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов (рис. 1), (рис. 2).

A screenshot of a text editor window. The window has a toolbar at the top with icons for opening, saving, and closing files, and a 'Save' button. The text inside the editor is a shell script. The script starts with a shebang line, then defines a lockfile path. It uses a while loop with a test command to check if the lockfile exists. Inside the loop, there's an if statement using flock to attempt to acquire a non-blocking lock. If successful, it prints 'File is blocked', sleeps for 5 seconds, prints 'File is unlocked', and releases the lock. If the lock is not acquired, it prints 'File is blocked' and sleeps for 5 seconds. The loop ends with 'fi' and 'done'.

```
#!/bin/bash

lockfile="./lock.file"
exec {fn}>$lockfile

while test -f "$lockfile"
do
if flock -n ${fn}
then
    echo "File is_blocked"
    sleep 5
    echo "File is unlocked"
    flock -u ${fn}
else
    echo "File is blocked"
    sleep 5
fi
done
```

Рис. 4.1: lab14-1.sh

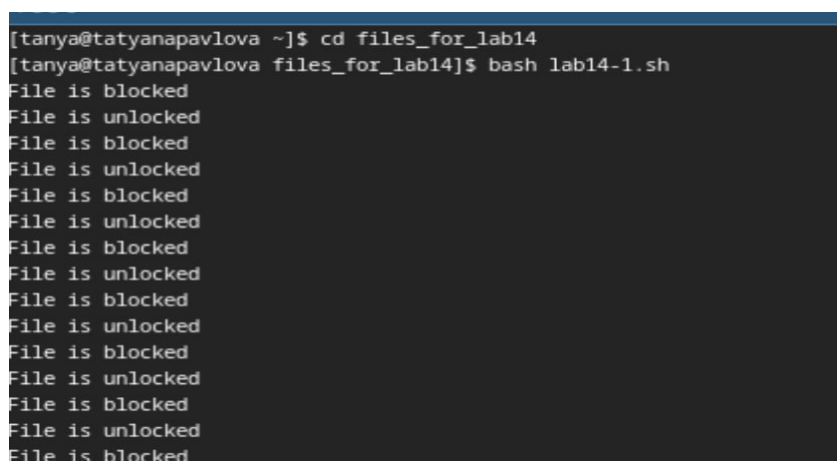
A screenshot of a terminal window. The prompt is [tanya@tatyana-pavlova ~]. The user enters 'cd files_for_lab14'. The prompt changes to [tanya@tatyana-pavlova files_for_lab14]. The user enters 'bash lab14-1.sh'. The script then runs, printing a series of 'File is blocked' and 'File is unlocked' messages. The output shows a sequence of 15 lines: 'File is blocked', 'File is unlocked', 'File is blocked', 'File is unlocked', 'File is blocked', 'File is unlocked', 'File is blocked', 'File is unlocked', 'File is blocked', 'File is unlocked', 'File is blocked', 'File is unlocked', 'File is blocked', 'File is unlocked', 'File is blocked'.

Рис. 4.2: Компиляция файла

Реализовать команду man с помощью командного файла. Изучите содержимое каталога /usr/share/man/man1. В нем находятся архивы текстовых файлов, со-

держащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой `less` сразу же просмотрев содержимое справки. Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1` (рис. 3), (рис. 4), (рис. 5), (рис. 6).

```
tanya@tatyapavlova ~]$ ls /usr/share/man/man1
.:1.gz
[.1.gz
7z.1.gz
c2ping.1.gz
abrt.1.gz
abrt-action-analyze-backtrace.1.gz
abrt-action-analyze-c.1.gz
abrt-action-analyze-ccpp-local.1.gz
abrt-action-analyze-oops.1.gz
abrt-action-analyze-python.1.gz
abrt-action-analyze-vmcore.1.gz
abrt-action-analyze-vulnerability.1.gz
abrt-action-analyze-xorg.1.gz
abrt-action-check-oops-for-fw-error.1.gz
abrt-action-find-bodhi-update.1.gz
abrt-action-generate-backtrace.1.gz
abrt-action-generate-core-backtrace.1.gz
abrt-action-list-dsos.1.gz
abrt-action-notify.1.gz
abrt-action-save-package-data.1.gz
abrt-action-trim-files.1.gz
abrt-applet.1.gz
abrt-auto-reporting.1.gz
abrt-bodhi.1.gz
abrt-cli.1.gz
abrt-dump-journal-core.1.gz
abrt-dump-journal-oops.1.gz
abrt-dump-journal-xorg.1.gz
abrt-dump-oops.1.gz
abrt-dump-xorg.1.gz
abrt-handle-upload.1.gz
abrt-harvest-pstoreoops.1.gz
abrt-harvest-vmcore.1.gz
abrt-merge-pstoreoops.1.gz
abrt-server.1.gz
abrt-watch-log.1.gz
msgmerge.1.gz
msgfmt.1.gz
msgunfmt.1.gz
msguniq.1.gz
ashortname.1.gz
ashowfat.1.gz
asxlint.1.gz
atools.1.gz
atoolstest.1.gz
atrace.1.gz
mtx-babel.1.gz
mtx-base.1.gz
mtx-bibtex.1.gz
mtx-cache.1.gz
mtx-chars.1.gz
mtx-check.1.gz
mtx-colors.1.gz
mtx-context.1.gz
mtx-dvi.1.gz
mtx-epub.1.gz
mtx-evohome.1.gz
mtx-fcd.1.gz
mtx-fcd.1.gz
mtx-flac.1.gz
mtx-fonts.1.gz
mtx-grep.1.gz
mtx-interface.1.gz
mtx-metapost.1.gz
mtx-modules.1.gz
mtx-package.1.gz
mtx-patterns.1.gz
mtx-pdf.1.gz
mtx-plain.1.gz
mtx-profile.1.gz
mtx-rsync.1.gz
mtxrun.1.gz
mtx-scite.1.gz
mtx-server.1.gz
```

Рис. 4.3: `ls /usr/share/man/man1`

```
#!/bin/bash

a=$1
if test -f "/usr/share/man/man1/$a.1.gz"
then less /usr/share/man/man1/$a.1.gz
else
echo "There is no such command"
fi
```

Рис. 4.4: `lab14-2.sh`


```
[tanya@tatyana-pavlova files_for_lab14]$ chmod +x lab14-3.sh  
[tanya@tatyana-pavlova files_for_lab14]$ bash lab14-3.sh 14  
lwovrhtcnlxupp
```

Рис. 4.8: Компиляция файла

5 Выводы

При выполнении данной лабораторной работы, я изучила основы программирования в оболочке ОС UNIX, а также научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.