

Лабораторная работа №12

Ханина Людмила Константиновна

Table of Contents

Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Задание

- 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 имела возможность взаимодействия трёх и более процессов.
- 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`.
- 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.

Выполнение лабораторной работы

1. Создадим файл `lab121.sh` и запишем в него скрипт, реализующий упрощённый механизм семафоров. Изменим доступ к файлу, чтобы можно было его запускать.

```
[lkkhanina@fedora ~]$ touch lab121.sh
[lkkhanina@fedora ~]$ vi lab121.sh
[lkkhanina@fedora ~]$ chmod +x lab121.sh
[lkkhanina@fedora ~]$ cat lab121.sh
#!/bin/bash
function run {
    s1=$(date + "%s")
    s2=$(date + "%s")
    ((t=$s2-$s1))
    while ((t < t2))
    do
        echo "Run"
        sleep 1
        s2=$(date + "%s")
        ((t=$s2-$s1))
    done
}

function wait {
    s1=$(date + "%s")
    s2=$(date + "%s")
    ((t=$s2-$s1))
    while ((t < t1))
    do
        echo "Wait"
        sleep 1
        s2=$(date + "%s")
        ((t=$s2-$s1))
    done
}

t1=$1
t2=$2
com=$3

while true
do
    if ["$com" == "Exit"]
    then
        echo "Exit"
        exit 0
    fi
    if ["$com" == "Wait"]
    then wait
    fi
    if ["$com" == "Run"]
    then run
    fi
    echo "Next action: "
    read com
done
[lkkhanina@fedora ~]$
```

Первый скрипт

2. Проверим, что он работает. Запустим.

```
[lkkhanina@fedora ~]$ ./lab121.sh 1 3 Wait > /dev/pts/1 &
[1] 2696
bash: /dev/pts/1: Отказано в доступе
[1]+  Выход 1                  ./lab121.sh 1 3 Wait > /dev/pts/1
[lkkhanina@fedora ~]$ ./lab121.sh 1 3 Wait > /dev/pts/2 &
[1] 2706
bash: /dev/pts/2: Отказано в доступе
[1]+  Выход 1                  ./lab121.sh 1 3 Wait > /dev/pts/2
[lkkhanina@fedora ~]$ ./lab121.sh 3 6 Run > /dev/pts/2 &
[1] 2715
bash: /dev/pts/2: Отказано в доступе
[1]+  Выход 1                  ./lab121.sh 3 6 Run > /dev/pts/2
[lkkhanina@fedora ~]$ ./lab121.sh 3 6 Run > /dev/pts/1 &
[1] 2728
bash: /dev/pts/1: Отказано в доступе
[1]+  Выход 1                  ./lab121.sh 3 6 Run > /dev/pts/1
[lkkhanina@fedora ~]$
```

Запускаем первый скрипт

3. Далее посмотрим на содержимое файла `/usr/share/man/man1`.

```
lkkhanina@fedora: ~]$ ls /usr/share/man/man1
:~.1.gz
'~.1.gz'
ab~.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-core~.1.gz
abrt-action-analyze-java~.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-hw-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-install-debuginfo~.1.gz
abrt-action-list-dsos~.1.gz
abrt-action-notify~.1.gz
abrt-action-perform-ccpp-analysis~.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
```

Содержимое файла /usr/share/man/man1

4. Далее создадим файл lab122.sh. Запишем скрипт, который будет искать введенное слово в каталоге /usr/share/man/man1 и выводить содержимое, то есть справку о команде. Изменим доступ к файлу lab122.sh, чтобы можно было его запускать.

```
[lkkhanina@fedora ~]$ vi lab122.sh
[lkkhanina@fedora ~]$ chmod +x lab122.sh
[lkkhanina@fedora ~]$ cat lab122.sh
#!/bin/bash
c=$1
if [ -f "/usr/share/man/man1/$c.1.gz" ]; then
    gunzip -c /usr/share/man/man1/$c.1.gz | less
else
    echo "Not found"
fi
[lkkhanina@fedora ~]$
```

Второй скрипт

5. Теперь запустим файл и увидим, что программа отработала корректно.

```
[lkkhanina@fedora ~]$
[lkkhanina@fedora ~]$ ./lab122.sh ldf
Not found
[lkkhanina@fedora ~]$ ./lab122.sh yes
```

Запускаем второй скрипт часть 1


```

.\" DO NOT MODIFY THIS FILE!  It was generated by help2man 1.47.3.
.TH YES "1" "July 2021" "GNU coreutils 8.32" "User Commands"
.SH NAME
yes \- output a string repeatedly until killed
.SH SYNOPSIS
.B yes
[\\fI\\,STRING\\fR]...
.br
.B yes
\\fI\\,OPTION\\fR
.SH DESCRIPTION
.\" Add any additional description here
.PP
Repeatedly output a line with all specified STRING(s), or 'y'.
.TP
\\fB\\-\\-help\\fR
display this help and exit
.TP
\\fB\\-\\-version\\fR
output version information and exit
.SH AUTHOR
Written by David MacKenzie.
.SH "REPORTING BUGS"
GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
.br
Report any translation bugs to <https://translationproject.org/team/>
.SH COPYRIGHT
Copyright \\(co 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.htm>
.br
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
.SH "SEE ALSO"
Full documentation <https://www.gnu.org/software/coreutils/yes>
:

```

Запускаем второй скрипт часть 2

6. Создадим файл lab123.sh и запишем в него скрипт, генерирующий случайную последовательность букв латинского алфавита.

```
[lkkhanina@fedora ~]$ vi lab123.sh
[lkkhanina@fedora ~]$ chmod +x lab123.sh
[lkkhanina@fedora ~]$ cat lab123.sh
#!/bin/bash
c=$1
for (( i=0; i<$c; i++ ))
do
    (( char=$RANDOM%26+1 ))
    case $char in
        1) echo -n a;;
        2) echo -n b;;
        3) echo -n c;;
        4) echo -n d;;
        5) echo -n e;;
        6) echo -n f;;
        7) echo -n g;;
        8) echo -n h;;
        9) echo -n i;;
        10) echo -n j;;
        11) echo -n k;;
        12) echo -n l;;
        13) echo -n m;;
        14) echo -n n;;
        15) echo -n o;;
        16) echo -n p;;
        17) echo -n q;;
        18) echo -n r;;
        19) echo -n s;;
        20) echo -n t;;
        21) echo -n u;;
        22) echo -n v;;
        23) echo -n w;;
        24) echo -n x;;
        25) echo -n y;;
        26) echo -n z
    esac
done
echo
```

Третий скринт

7. Теперь запустим файл и увидим, что программа отработала корректно.

```
[lkkhanina@fedora ~]$ ./lab123.sh 5
btzmn
[lkkhanina@fedora ~]$ ./lab123.sh 17
oeximvjazzukalvyj
[lkkhanina@fedora ~]$ ./lab123.sh 165
bbghvdqumuehpmiqzruawxzcabyzttigtapemojqyirlmkfvthydnvwxgajhlaakgvvlppnhvknnfk
bdjxozhynfjxyvtltndqbopmiblqgdkdofcjmmezwcacpzvkdqczobjxnyryotnokqycmldvodoLxvj
fevrw
[lkkhanina@fedora ~]$
```

Запускаем третий скрипт

Контрольные вопросы

1. while [\$1 != "exit"]

В данной строчке допущены следующие ошибки:

- не хватает пробелов после первой скобки и перед второй скобкой;
- выражение \$1 необходимо взять в "".

Таким образом, правильный вариант должен выглядеть так: while ["\$1" != "exit"]

- Для объединения строк в одну можно воспользоваться следующими способами:
 - newstring="s1s2"
 - string += "text"
- Утилита seq выводит последовательность целых чисел с шагом, заданным пользователем.
 - Чтобы просто напечатать последовательность чисел, начиная с 1 и заканчивая n, используйте следующую команду: seq n;
 - Можно указать верхний и нижний пределы: seq begin end (seq 1 3 -> 1 2 3);
 - Также можно указать шаг: seq begin step end (seq 1 3 10 -> 1 4 7 10).
- Ответом будет 3, так как было использовано целочисленное деление.
- Основные ключевые различия между Zsh и Bash:
 - Zsh более интерактивный и настраиваемый, чем Bash.
 - У Zsh есть поддержка с плавающей точкой, которой нет у Bash.
 - В Zsh поддерживаются структуры хеш-данных, которых нет в Bash.
 - Функции вызова в Bash лучше по сравнению с Zsh.
 - Внешний вид подсказки можно контролировать в Bash, тогда как Zsh настраивается.
 - Конфигурационными файлами являются .bashrc в интерактивных оболочках без регистрации и .profile или .bash_profile в оболочках входа в Bash. В Zsh оболочками, не входящими в систему, являются .zshrc, а оболочками для входа - .zprofile.
 - Массивы Zsh индексируются от 1 до длины, тогда как Bash индексируется от -1 до длины.
 - В Zsh, если шаблоны не совпадают ни с одним файлом, выдается ошибка. Находясь в Баше, он остается без изменений.

6. Синтаксис верен.

7. Преимущества sh:

- стандартизировано;
- это намного проще и легче узнать;
- он переносится через системы POSIX - даже если они не имеют bash, они должны иметь sh.

Есть и преимущества использования bash. Его функции делают программирование более удобным и похожим на программирование на других современных языках программирования. К ним относятся такие области, как локальные переменные и массивы. Обычный sh - очень минималистический язык программирования.

Выводы

Я научилась писать более сложные командный файлы с использованием логических управляющих конструкций и циклов.