

Лабораторная работа №12

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Рогожина Н.А.

13 апреля 2023

Российский университет дружбы народов, Москва, Россия

Информация

- Рогожина Надежда Александровна
- Студентка 1го курса, НКАбд-02-22
- Компьютерные и информационные науки
- Российский университет дружбы народов
- Github

Вводная часть

- Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.

- Командный процессор

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

Содержание лабораторной работы

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 оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;

- оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

Задание

Задачи:

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.

Задачи:

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.

Рис. 1: Скрипт №1

```
[narogozhina@narogozhina labs]$ cd  
[narogozhina@narogozhina ~]$ emacs script1  
[narogozhina@narogozhina ~]$ bash script1  
The process is blocked  
Process is blocking the resource  
Until unblocking: 4 sec.  
Until unblocking: 3 sec.  
Until unblocking: 2 sec.  
Until unblocking: 1 sec.  
Resource is unblocked, getting out.  
[narogozhina@narogozhina ~]$
```

Рис. 2: Выполнение скрипта №1

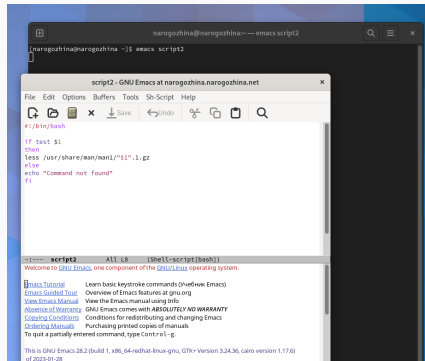
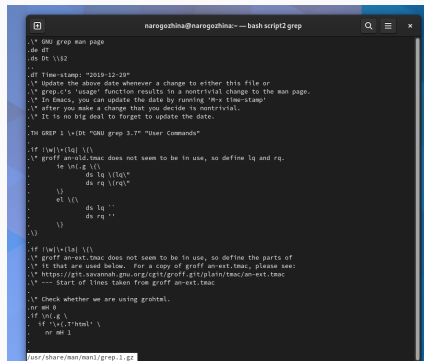
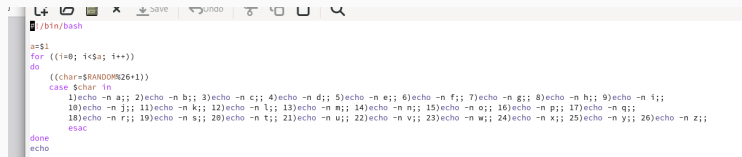


Рис. 3: Реализация команды man



```
narogozhina@narogozhina:~ --- bash script2 grep
.* GNU grep man page
.de dt
.ds Dt \\$2
..
.dT Time-stamp: "2019-12-20"
.* Update the above date whenever a change to either this file or
.* grep.c's 'usage' function results in a nontrivial change to the man page.
.* In Emacs, you can update the date by running 'M-x time-stamp'
.* after you make a change that you decide is nontrivial.
.* It is no big deal to forget to update the date.
.
.TH GREP 1 \"(Dt \"GNU grep 3.7\" \"User Commands\"
.
.if \\w|*(lq) \\(\\
.* groff an-ld.tmac does not seem to be in use, so define lq and rq.
.  te \\n(.g \\(\\
.    ds lq \\(lq\\*
.    ds rq \\(rq\\*
.  \\}
.  el \\(\\
.    ds lq ''
.    ds rq ''
.  \\}
.\\)
.
.if \\w|*(la) \\(\\
.* groff an-ext.tmac does not seem to be in use, so define the parts of
.* it that are used below.  For a copy of groff an-ext.tmac, please see:
.* https://git.sv.gnu.org/git/groff.git/plain/tmac/an-ext.tmac
.* --- Start of lines taken from groff an-ext.tmac
.
.* Check whether we are using grohtml.
.nr mh 0
.if \\n(.g \\
.  if '\\n(.T'html' \\
.    nr mh 1
.
./usr/share/man/man1/grep.1.gz
```

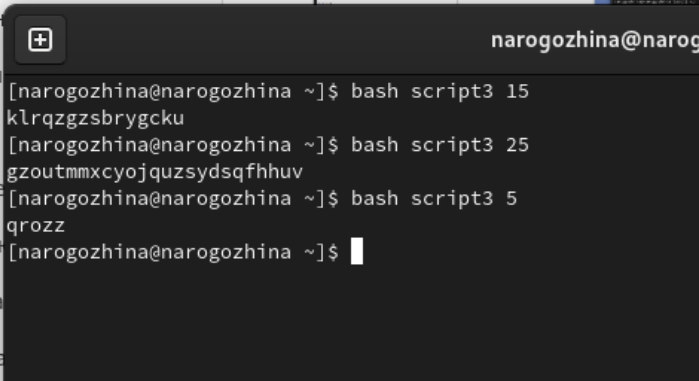
Рис. 4: Пример использования



```
#!/bin/bash

a=$1
for ((i=0; i<$a; 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
```

Рис. 5: Скрипт №3



A terminal window titled 'narogozhina@narog' with a '+' icon in the top-left corner. The terminal shows the execution of a script named 'script3' with three different arguments: 15, 25, and 5. Each execution produces a long alphanumeric string. The prompt is '[narogozhina@narogozhina ~]\$'.

```
[narogozhina@narogozhina ~]$ bash script3 15
klrqzgzsbyrgcku
[narogozhina@narogozhina ~]$ bash script3 25
gzoutmmxcyojqzsydsqfhuv
[narogozhina@narogozhina ~]$ bash script3 5
qrozz
[narogozhina@narogozhina ~]$
```

Рис. 6: Примеры выполнения

Выводы

В ходе лабораторной работы мы научились писать командные файлы с использованием логических управляющих конструкций и циклов.