

Российский университет дружбы народов

факультет физико-математических и естественных наук

Отчет по лабораторной работе № 13

дисциплина : Операционные системы

студент Блохин александр НКН

Москва

2021

Цель работы

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

Ход работы

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 имела возможность взаимодействия трёх и более процессов.

```
#!/bin/bash
lockfile = "./lock.file"
exec {fn}>lockfile

echo "locked"
until flock -n ${fn}
do.
echo "Not locked"
sleep 1
flock -n ${fn}
done
for ((i=0;i<=7;i++))
do.
echo "work"
sleep 1
done
```

```
al@al-VirtualBox:~$ sh l13.sh
Locked
work
work
work
work
work
work
work
work
```

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`.

```
emacs@al
File Edit Options Buffers Tools Text Help
[Icons] Save Undo
cd /usr/share/mam/ma1
if (test -f $1.1.gz)
then less $1.1.gz
else echo "non file"
fi
```

```
al@al-VirtualBox:~$ emacs
al@al-VirtualBox:~$ sh r1.txt cdd
r1.txt: 1: cd: can't cd to /usr/share/mam/ma1
non file
al@al-VirtualBox:~$ sh r1.txt lls
r1.txt: 1: cd: can't cd to /usr/share/mam/ma1
non file
al@al-VirtualBox:~$
```

```
ls - list directory contents

SYNOPSIS
ls [OPTION]... [FILE]...

DESCRIPTION
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is speci-
fied.

Mandatory arguments to long options are mandatory for short options
too.

-a, --all
do not ignore entries starting with .

Ubuntu Software
-m, --most-all
do not list implied . and ..

--author
with -l, print the author of each file

-b, --escape
print C-style escapes for nongraphic characters
```

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.

```
letters=( a b c d e f g h i j k l m n o p q r s t u v w x y z)
let b=$RANDOM%26
i=0
while (( i < b ))
do
let k=$RANDOM%26
echo -n ${letters[$k]}
let i+=1
done
```

```
al@al-VirtualBox:~$ emacs
al@al-VirtualBox:~$ bash random.txt
tdasvbnfvfpngversarqzval@al-VirtualBox:~$ bash random.txt
kial@al-VirtualBox:~$ bash random.txt
ckrsybvykznal@al-VirtualBox:~$ bash random.txt
rsnmxfionshsjbxasxhydfmal@al-VirtualBox:~$ bash random.txt
bxwkaajvfsxuhrejuval@al-VirtualBox:~$
```

Вывод

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

контрольные вопросы

1. while ["\$1" != "exit"]
2. С помощью знака >| можно объединить несколько строк в одну.
3. Эта утилита выводит последовательность целых чисел с заданным шагом. Также можно реализовать с помощью утилиты jot.
4. Результатом вычисления выражения $\$(10/3)$ будет число 3.
5. В zsh можно настроить отдельные сочетания клавиш так, как вам нравится. Использование истории команд в zsh ничем особенным не отличается от bash. Zsh очень удобен для повседневной работы и делает добрую половину рутины за вас. Но стоит обратить внимание на различия между этими двумя оболочками. Например, в zsh после for обязательно вставлять пробел, нумерация массивов в zsh начинается с 1, чего совершенно невозможно понять. Так, если вы используете shell для повседневной работы, исключающей написание скриптов, используйте zsh. Если вам часто приходится писать свои скрипты, только bash! Впрочем, можно комбинировать.
6. Конструкция for ((a=1; a <= LIMIT; a++)) верена.
7. в основном cpp "всего" в 2 раза быстрее, чем bash, а python намного медленнее.