

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 14__

дисциплина: *Операционные системы*

Студент: Ниemek Яи Жак

Группа: НММБд-04-24

МОСКВА

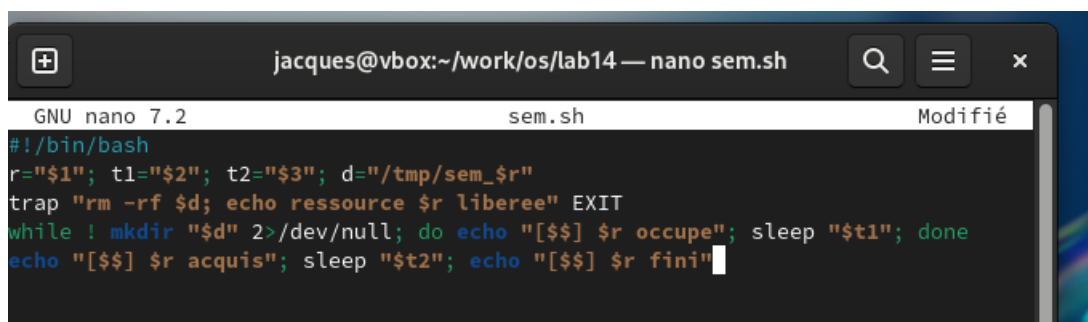
2025__ г.

Цель работы

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

Последовательность выполнения 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.
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



```
jacques@vbox:~/work/os/lab14 — nano sem.sh
GNU nano 7.2 sem.sh Modifié
#!/bin/bash
r="$1"; t1="$2"; t2="$3"; d="/tmp/sem_$r"
trap "rm -rf $d; echo ressource $r liberee" EXIT
while ! mkdir "$d" 2>/dev/null; do echo "[$$] $r occupe"; sleep "$t1"; done
echo "[$$] $r acquis"; sleep "$t2"; echo "[$$] $r fini"
```

```

jacques@vbox:~/work/os/lab14$ touch sem.sh
jacques@vbox:~/work/os/lab14$ nano sem.sh
jacques@vbox:~/work/os/lab14$ chmod +x sem.sh
jacques@vbox:~/work/os/lab14$ ./sem.sh
[13475] acquis
sleep: intervalle de temps «  » incorrect
Saisissez « sleep --help » pour plus d'informations.
[13475] fini
ressource liberee
jacques@vbox:~/work/os/lab14$ ./sem.sh printer 2 5
[13487] printer acquis
[13487] printer fini
ressource printer liberee
jacques@vbox:~/work/os/lab14$

```

```

GNU nano 7.2 myman.sh
#!/bin/bash

f="/usr/share/man/man1/$1.1"
[[ -f $f ]] || f="$f.gz"
[[ -f $f ]] || { echo "pas de man"; exit 1; }
case $f in *.gz) gzip -cd "$f" | less;; *) less "$f";; esac

```

```

+ jacques@vbox:~/work/os/lab14 — /bin/bash ./myman.sh ls

.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.48.5.
.TH LS "1" "September 2023" "GNU coreutils 9.3" "User Commands"
.SH NAME
ls \- list directory contents
.SH SYNOPSIS
.B ls
[ \fI\,OPTION\ \fR ]... [ \fI\,FILE\ \fR ]...
.SH DESCRIPTION
.\" Add any additional description here
.PP
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of \fB\-\cftuvSUX\fR nor \fB\-\-sort\fR is specified.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
\fB\-a\fR, \fB\-\-all\fR
do not ignore entries starting with .
.TP
\fB\-A\fR, \fB\-\-almost-all\fR
do not list implied . and ..
.TP
\fB\-\-author\fR
with \fB\-l\fR, print the author of each file

```

```

jacques@vbox:~/work/os/lab14$ touch myman.sh
jacques@vbox:~/work/os/lab14$ nano myman.sh
jacques@vbox:~/work/os/lab14$ chmod +x myman.sh
jacques@vbox:~/work/os/lab14$ ./myman.s ls
bash: ./myman.s: Aucun fichier ou dossier de ce type
jacques@vbox:~/work/os/lab14$ ./myman.sh ls
jacques@vbox:~/work/os/lab14$

```

```

+ jacques@vbox:~/work/os/lab14 — nano random_letters.sh
GNU nano 7.2 random_letters.sh
#!/bin/bash
n="$1"; m="${2:---lower}"
for((i=0;i<n;i++));do
    case "$m" in
        --lower) printf \\$(printf '%03o' $((97+RANDOM%26)));;
        --upper) printf \\$(printf '%03o' $((65+RANDOM%26)));;
        --mixed) printf \\$(printf '%03o' $(( (RANDOM%2==0?97:65)+RANDOM%26 )));;
    esac
done; echo
```

```

jacques@vbox:~/work/os/lab14$ touch random_letters.sh
jacques@vbox:~/work/os/lab14$ nano random_letters.sh
jacques@vbox:~/work/os/lab14$ chmod +x random_letters.sh
jacques@vbox:~/work/os/lab14$ ./random_letters.sh 16 --mixed
VGfgLHiiqqlyZaBV
jacques@vbox:~/work/os/lab14$ ./random_letters.sh 16 --lower
mafrmwkrpoyafqod
jacques@vbox:~/work/os/lab14$
```

1. Синтаксическая ошибка в while [\$1 != "exit"]

Ошибка в отсутствии пробелов вокруг [и]. Правильный вариант:

```
while [ "$1" != "exit" ]; do
    # команды
done
```

* Пробелы обязательны после [и перед].

* Лучше всегда брать переменные в кавычки (" \$1 "), чтобы избежать ошибок при пустом значении.

2. Конкатенация нескольких строк

* В Bash строки объединяются простым объединением переменных:

```
str1="Hello"
str2="World"
result="$str1 $str2"    # результат: "Hello World"
```

* Можно использовать += для добавления:

```
str="Hello"
str+=" World"
```

3. Утилита seq и альтернативы

* seq генерирует последовательность чисел:

```
seq 1 5    # 1 2 3 4 5
```

* Альтернативы в Bash:

```
for i in {1..5}; do echo $i; done
```

```
for ((i=1; i<=5; i++)); do echo $i; done
```

4. Результат выражения $\$(10/3)$

* В Bash выполняется целочисленное деление, дробная часть отбрасывается.

```
echo  $\$(10/3)$     # результат: 3
```

5. Основные отличия Zsh от Bash

Особенность	Bash	Zsh
Автодополнение	ограниченное	умное, с меню и подсказками
Глобальные алиасы	нет	есть
Массивы	индекс с 0	можно с 1 или 0
Промпт	простой	расширенный, легко настраивается
История	стандартная	расширенная, удобный поиск

6. Синтаксис `for ((a=1; a <= LIMIT; a++))`

* Конструкция правильная только с ``do ... done``:

```
for ((a=1; a <= LIMIT; a++)); do
```

```
    echo $a
```

```
done
```

* Без `do ... done` Bash выдаст синтаксическую ошибку.

7. Сравнение Bash с другими языками

Преимущества Bash:

* Простота автоматизации задач в Unix/Linux

- * Работа с файлами, потоками и процессами «из коробки»
- * Легко писать скрипты для системного администрирования

Недостатки:

- * Медленнее, чем Python или C для вычислений
- * Сложнее отлаживать сложные скрипты
- * Ограниченные структуры данных (только массивы и строки)