

Архитектура компьютера Отчёт по лабораторной работе №14

Лю Сяо НКАбд-04-24

Цель работы

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

Результаты выполнения задания

Задание 1: Упрощённый механизм семафоров

```
liveuser@localhost-live:~/work/lab14$ nano semaphore.sh
liveuser@localhost-live:~/work/lab14$ cat semaphore.sh
#!/bin/bash

LOCK_FILE="/tmp/resource.lock"
WAIT_TIME=$1
USE_TIME=$2

echo "Процесс $$ ожидает освобождения ресурса..."

while [ -f "$LOCK_FILE" ] && [ $WAIT_TIME -gt 0 ]; do
    sleep 1
    WAIT_TIME=$((WAIT_TIME-1))
    echo "Процесс $$: ресурс занят, осталось времени ожидания: $WAIT_TIME сек"
done

if [ $WAIT_TIME -le 0 ]; then
    echo "Процесс $$: время ожидания истекло, выход"
    exit 1
fi

touch "$LOCK_FILE"
echo "Процесс $$: ресурс получен, использование в течение $USE_TIME сек"
sleep $USE_TIME
rm -f "$LOCK_FILE"
echo "Процесс $$: ресурс освобожден"
liveuser@localhost-live:~/work/lab14$
```

Листинг программы:

```
#!/bin/bash

LOCK_FILE="/tmp/resource.lock"
WAIT_TIME=$1
USE_TIME=$2
```

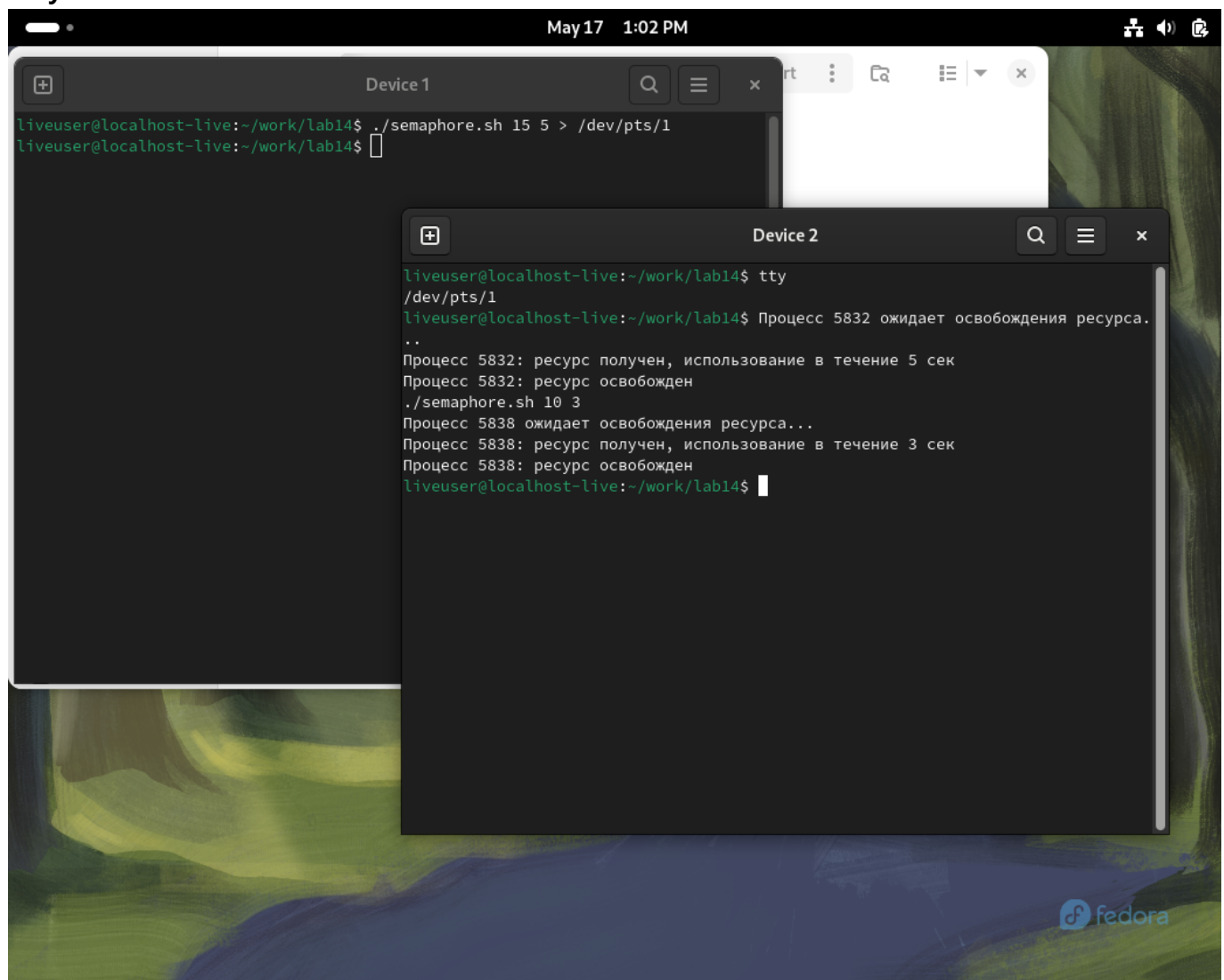
```
echo "Процесс $$ ожидает освобождения ресурса..."

while [ -f "$LOCK_FILE" ] && [ $WAIT_TIME -gt 0 ]; do
    sleep 1
    WAIT_TIME=$((WAIT_TIME-1))
    echo "Процесс $$: ресурс занят, осталось времени ожидания: $WAIT_TIME сек"
done

if [ $WAIT_TIME -le 0 ]; then
    echo "Процесс $$: время ожидания истекло, выход"
    exit 1
fi

touch "$LOCK_FILE"
echo "Процесс $$: ресурс получен, использование в течение $USE_TIME сек"
sleep $USE_TIME
rm -f "$LOCK_FILE"
echo "Процесс $$: ресурс освобожден"
```

Результаты теста:



```
Device 1
liveuser@localhost-live:~/work/lab14$ ./semaphore.sh 15 5 > /dev/pts/1
liveuser@localhost-live:~/work/lab14$

Device 2
liveuser@localhost-live:~/work/lab14$ tty
/dev/pts/1
liveuser@localhost-live:~/work/lab14$ Процесс 5832 ожидает освобождения ресурса.
..
Процесс 5832: ресурс получен, использование в течение 5 сек
Процесс 5832: ресурс освобожден
./semaphore.sh 10 3
Процесс 5838 ожидает освобождения ресурса...
Процесс 5838: ресурс получен, использование в течение 3 сек
Процесс 5838: ресурс освобожден
liveuser@localhost-live:~/work/lab14$
```

Задание 2: Реализация команды map

```
liveuser@localhost-live:~/work/lab14$ nano my_man.sh
liveuser@localhost-live:~/work/lab14$ cat my_man.sh
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Использование: ./my_man.sh <команда>"
    exit 1
fi

MAN_FILE="/usr/share/man/man1/$1.1.gz"

if [ -f "$MAN_FILE" ]; then
    less "$MAN_FILE"
else
    echo "Справка по команде '$1' не найдена"
fi
liveuser@localhost-live:~/work/lab14$
```

Листинг программы:

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Использование: ./my_man.sh <команда>"
    exit 1
fi

MAN_FILE="/usr/share/man/man1/$1.1.gz"

if [ -f "$MAN_FILE" ]; then
    less "$MAN_FILE"
else
    echo "Справка по команде '$1' не найдена"
fi
```

Результаты выполнения:

```

ESC[1mDESCRIPTIONESC[0m
    List information about the FILES (the current directory by default).
    Sort entries alphabetically if none of ESC[1m-cftuvSUX ESC[22mnor ESC[1m-
-sort ESC[22mis speci-
    fied.

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

    ESC[1m-aESC[22m, ESC[1m--allESC[0m
        do not ignore entries starting with .

    ESC[1m-AESC[22m, ESC[1m--almost-allESC[0m
        do not list implied . and ..

    ESC[1m--authorESC[0m
        with ESC[1m-lESC[22m, print the author of each file

    ESC[1m-bESC[22m, ESC[1m--escapeESC[0m
        print C-style escapes for nongraphic characters

    ESC[1m--block-sizeESC[22m=ESC[4mSIZEESC[0m
        with ESC[1m-lESC[22m, scale sizes by SIZE when printing th
em; e.g.,
        '--block-size=M'; see SIZE format below

:

```

Проверьте, что команда не существует::

```

liveuser@localhost-live:~/work/lab14$ ./my_man.sh ls
liveuser@localhost-live:~/work/lab14$ ./my_man.sh nonexistent_cmd
Справка по команде 'nonexistent_cmd' не найдена
liveuser@localhost-live:~/work/lab14$

```

Задание 3: Генерация случайной последовательности букв

```

liveuser@localhost-live:~/work/lab14$ nano random_letters.sh
liveuser@localhost-live:~/work/lab14$ cat random_letters.sh
#!/bin/bash

LENGTH=${1:-10} # Длина последовательности, по умолчанию 10

for ((i=0; i<LENGTH; i++)); do
    # Генерация случайного числа в диапазоне 0-25 (A-Z)
    RAND=$((RANDOM % 26))
    # Преобразование в букву (A=65 в ASCII)
    LETTER=$(printf '\$(printf '%03o' $((RAND + 65)))')
    echo -n $LETTER
done
echo
liveuser@localhost-live:~/work/lab14$

```

Листинг программы:

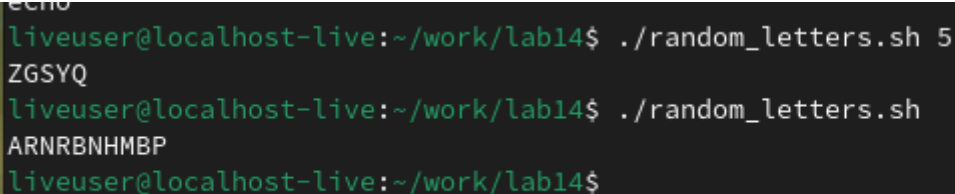
```
#!/bin/bash

LENGTH=${1:-10} # Длина последовательности, по умолчанию 10

for ((i=0; i<LENGTH; i++)); do
    # Генерация случайного числа в диапазоне 0-25 (A-Z)
    RAND=$((RANDOM % 26))
    # Преобразование в букву (A=65 в ASCII)
    LETTER=$(printf \\$(printf '%03o' $((RAND + 65))))
    echo -n $LETTER
done
echo
```

Результаты выполнения:

```
./random_letters.sh 5
./random_letters.sh
```



```
liveuser@localhost-live:~/work/lab14$ ./random_letters.sh 5
ZGSYQ
liveuser@localhost-live:~/work/lab14$ ./random_letters.sh
ARNRBNHMBP
liveuser@localhost-live:~/work/lab14$
```

Ответы на контрольные вопросы

1. Синтаксическая ошибка: отсутствуют пробелы внутри квадратных скобок. Правильный вариант:

```
while [ "$1" != "exit" ]
```

2. Конкатенация строк в bash:

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

3. Утилита `seq` генерирует последовательность чисел. Альтернативы в bash:

```
# С использованием фигурных скобок
echo {1..10}
# С использованием цикла
for i in $(seq 1 10); do echo $i; done
```

4. Результат выражения $\$(10/3)$ будет 3 (целочисленное деление).

5. Основные отличия zsh от bash:

- Более развитое автодополнение
- Расширенные возможности подстановки
- Лучшая обработка массивов
- Поддержка плавающей арифметики

6. Синтаксис конструкции верен. Это стандартный цикл for в стиле C.

7. Сравнение bash с другими языками:

- Преимущества: простота работы с процессами и файлами, встроенная в ОС
- Недостатки: медленнее компилируемых языков, менее строгий синтаксис

Выводы

В ходе лабораторной работы были изучены основы программирования в командном процессоре UNIX. Были освоены методы работы с управляющими конструкциями, циклами и системными вызовами. Приобретены навыки создания более сложных командных файлов, включая реализацию механизма семафоров, аналога команды map и генератора случайных последовательностей. Полученные знания позволяют эффективно автоматизировать задачи в среде UNIX.