

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине «Операционная система Linux»

Управление процессами ОС Ubuntu

Студент

Мастылина А.А.

Группа АИ-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

Оглавление

Цель работы	3
Задание кафедры.....	4
Ход работы	5
Вывод	12
Контрольные вопросы.....	13

Цель работы

Знакомство со средами управления процессами ОС Ubuntu.

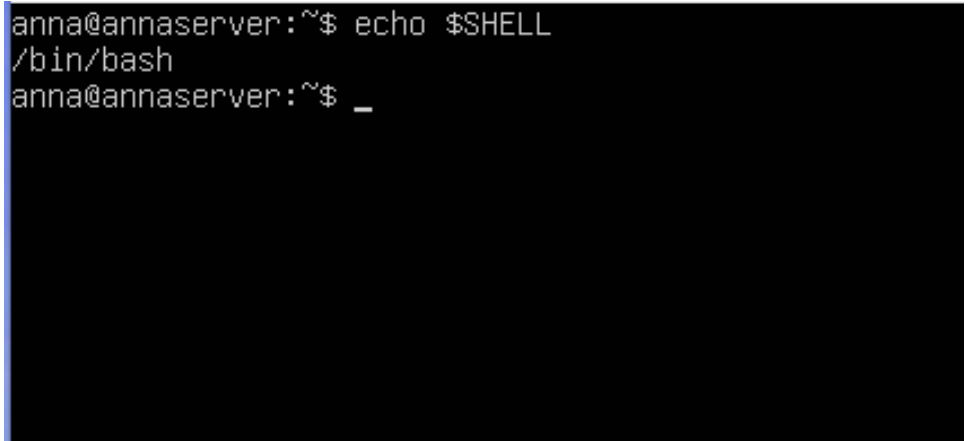
Задание кафедры

1. Запустить программу виртуализации Oracle VM VirtualBox
2. Запустить виртуальную машину Ubuntu.
3. Открыть окно интерпретатора команд
4. Вывести общую информацию о системе
 - 4.1 Вывести информацию о текущем интерпретаторе команд
 - 4.2 Вывести информацию о текущем пользователе
 - 4.3 Вывести информацию о текущем каталоге
 - 4.4 Вывести информацию об оперативной памяти и области подкачки
 - 4.5 Вывести информацию о дисковой памяти
5. Выполнить команды получения информации о процессах
 - 5.1 Получить идентификатор текущего процесса (PID)
 - 5.2 Получить идентификатор родительского процесса (PPID)
 - 5.3 Получить идентификатор процесса инициализации системы
 - 5.4 Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд
 - 5.5 Отобразить все процессы
6. Выполнить команды управления процессами
 - 6.1 Получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе
 - 6.2 Определить текущее значение nice по умолчанию
 - 6.2 Запустить интерпретатор bash с понижением приоритета nice -n 10
bash
 - 6.3 Определить PID запущенного интерпретатора
 - 6.4 Установить приоритет запущенного интерпретатора равным 5
renice -n 5 <PID процесса>
 - 6.5 Получить информацию о процессах bash
ps lax | grep bash

Ход работы

Запустим программу виртуализации Oracle VM VirtualBox и запустим виртуальную машину Ubuntu, откроем окно интерпретатора команд и выведем общую информацию о системе.

Выведем информацию о текущем интерпретаторе команд с помощью команды `echo $SHELL`. Результат работы представлен на рисунке 1.



```
anna@annaserver:~$ echo $SHELL
/bin/bash
anna@annaserver:~$ _
```

Рисунок 1 – Информация о текущем интерпретаторе команд

Выведем информацию о текущем пользователе с помощью команды `whoami`. Результат работы представлен на рисунке 2.



```
anna@annaserver:~$ whoami
anna
anna@annaserver:~$ _
```

Рисунок 2 - информацию о текущем пользователе

Выведем информацию о текущем каталоге с помощью команды `pwd`.
Результат работы представлен на рисунке 3.

```
anna@annaserver:~$ pwd
/home/anna
anna@annaserver:~$
```

Рисунок 3 - Информация о текущем каталоге

Выведем информацию об оперативной памяти и области подкачки с помощью команды `free`. Результат работы представлен на рисунке 4.

```
anna@annaserver:~$ free
              total        used         free       shared    buff/cache   available
Mem:           1004848        162304         484148           1036        3583396        690960
Swap:          1808380           0         1808380
anna@annaserver:~$ _
```

Рисунок 4 - Информация об оперативной памяти и области подкачки

Выведем информацию о дисковой памяти с помощью команды `df`.
Результат работы представлен на рисунке 5.

```
anna@annaserver:~$ df
Filesystem            1K-blocks    Used Available Use% Mounted on
udev                  458744         0    458744   0% /dev
tmpfs                 100488        1036    99452   2% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 9219412 4527216 4204160 52% /
tmpfs                 502424         0    502424   0% /dev/shm
tmpfs                  5120          0     5120   0% /run/lock
tmpfs                 502424         0    502424   0% /sys/fs/cgroup
/dev/loop1             56704        56704         0 100% /snap/core18/1932
/dev/sda2             999320      105468    825040 12% /boot
/dev/loop2             72320       72320         0 100% /snap/lxd/16922
/dev/loop4             31744       31744         0 100% /snap/snapd/9721
/dev/loop0             56320       56320         0 100% /snap/core18/1880
/dev/loop3             69376       69376         0 100% /snap/lxd/18150
/dev/loop5             30720       30720         0 100% /snap/snapd/8542
tmpfs                 100484         0    100484   0% /run/user/1000
anna@annaserver:~$ _
```

Рисунок 5 - Информация о дисковой памяти

Следующим заданием выполним команды получения информации о процессах

Получим идентификатор текущего процесса (PID) с помощью команды `echo $$`. Результат работы представлен на рисунке 6.

```
anna@annaserver:~$ echo $$  
973  
anna@annaserver:~$ _
```

Рисунок 6 - Идентификатор текущего процесса

Получим идентификатор родительского процесса (PPID). С помощью команды `echo $PPID`. Результат работы представлен на рисунке 7.

```
anna@annaserver:~$ echo $PPID  
613  
anna@annaserver:~$ _
```

Рисунок 7 - Идентификатор родительского процесса

Получим идентификатор процесса инициализации системы. С помощью команд `pidof bash`. Результат работы представлен на рисунке 8.

```
anna@annaserver:~$ pidof bash
973
anna@annaserver:~$ _
```

Рисунок 8 - Идентификатор процесса инициализации системы

Получим информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд. С помощью команды `ps`. Результат работы представлен на рисунке 9.

```
anna@annaserver:~$ ps
  PID TTY          TIME CMD
   973 tty1        00:00:00 bash
  1034 tty1        00:00:00 ps
anna@annaserver:~$
```

Рисунок 9 - Информация о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд

Отообразим все процессы с помощью команды `ps -e`. Результат работы представлен на рисунке 10.

```
368 ?      00:00:00 iprt-VBoxWQueue
488 ?      00:00:00 kaluad
489 ?      00:00:00 kmpath_rdadcd
490 ?      00:00:00 kmpathd
491 ?      00:00:00 kmpath_handlerd
492 ?      00:00:00 multipathd
503 ?      00:00:00 loop0
505 ?      00:00:00 loop1
508 ?      00:00:00 loop2
509 ?      00:00:00 loop3
510 ?      00:00:00 loop4
511 ?      00:00:00 loop5
512 ?      00:00:00 jbd2/sda2-8
513 ?      00:00:00 ext4-rsv-conver
525 ?      00:00:00 systemd-timesyn
566 ?      00:00:00 systemd-network
568 ?      00:00:00 systemd-resolve
582 ?      00:00:00 accounts-daemon
585 ?      00:00:00 cron
586 ?      00:00:00 dbus-daemon
595 ?      00:00:00 networkd-dispat
597 ?      00:00:00 rsyslogd
601 ?      00:00:01 snapd
605 ?      00:00:00 systemd-logind
609 ?      00:00:00 atd
613 tty1    00:00:00 login
637 ?      00:00:00 unattended-upgr
642 ?      00:00:01 fwupd
668 ?      00:00:00 polkitd
966 ?      00:00:00 systemd
968 ?      00:00:00 (sd-pam)
973 tty1    00:00:00 bash
1010 ?     00:00:02 kworker/0:1-events
1012 ?     00:00:00 kworker/u2:1-events_power_efficient
1031 ?     00:00:00 kworker/u2:0-events_unbound
1036 tty1    00:00:00 ps
anna@annaserver:~$
```

Рисунок 10 – Все процессы

Следующим заданием выполним команды управления процессами.

Получим информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе с помощью команды `ps`. Результат работы представлен на рисунке 11.

```
anna@annaserver:~$ ps
  PID TTY          TIME CMD
  973 tty1        00:00:00 bash
 1038 tty1        00:00:00 ps
anna@annaserver:~$ _
```

Рисунок 11 – Информация о выполняющихся процессах текущего пользователя в текущем интерпретаторе

Определим текущее значение `nice` по умолчанию с помощью команды `nice`. Результат работы представлен на рисунке 12.

```
anna@annaserver:~$ nice
0
anna@annaserver:~$ _
```

Рисунок 12 - Текущее значение `nice` по умолчанию

Запустим интерпретатор `bash` с понижением приоритета с помощью команды `nice -n 10 bash`. Результат работы представлен на рисунке 13.

```
anna@annaserver:~$ nice -n 10 bash
```

Рисунок 13 – Запуск `bash` с понижением приоритета

Определим PID запущенного интерпретатора. С помощью команды `echo $$`. Результат работы представлен на рисунке 14.

```
anna@annaserver:~$ echo $$
1042
```

Рисунок 14 – PID запущенного интерпретатора

Установим приоритет запущенного интерпретатора равным 5 с помощью команды `renice -n 5 <PID процесса>`. Результат работы представлен на рисунке 15.

```
anna@annaserver:~$ sudo renice -n 5 1042
1042 (process ID) old priority 10, new priority 5
anna@annaserver:~$
```

Рисунок 15 – Установка приоритета запущенного интерпретатора

Получим информацию о процессах bash с помощью команды `ps lax | grep bash`. Результат работы представлен на рисунке 16.

```
anna@annaserver:~$ ps lax | grep bash
4 1000  973    613 20  0 7068 5044 do_wai S   tty1      0:00 -bash
0 1000  1042    973 25  5 7036 4928 do_wai SN  tty1      0:00  bash
0 1000  1094    1042 25  5 5192  736 pipe_w SN+  tty1      0:00 grep --color=auto  bash
anna@annaserver:~$
```

Рисунок 16 – Информация о процессах bash

Вывод

В ходе выполнения лабораторной работы я ознакомилась со средами управления процессами ОС Ubuntu.

Контрольные вопросы

1. Перечислите состояния задачи в ОС Ubuntu

- 1) Running (выполнение) – после выделения ей процесса;
- 2) Sleeping (спячки) – состояние блокировки;
- 3) Stopped (останов) – остановка работы;
- 4) Zombie (зомби) – выполнение задачи прекратилось, однако она не была удалена;
- 5) Dead (смерть) – задача может быть удалена из системы;
- 6) Active (активный) и inspired (неактивный) – используются при планировании выполнения процесса.

2. Как создаются задачи в ОС Ubuntu

В системе Linux и процессы, и потоки называют задачами (task). Изнутри они представляют собой единую структуру данных. Планировщик процессов хранит список всех задач в виде двух структур данных. Первая структура представляет собой кольцевой список, каждая запись которого содержит указатели на предыдущую и последующую задачу. Обращение к этой структуре происходит в том случае, когда ядру необходимо проанализировать все задачи, которые должны быть выполнены в системе. Второй структурой является хэш-таблица. При создании задачи ей присваивается уникальный идентификатор процесса (process identifier, PID). Идентификаторы процессов передаются хэш-функции для определения местоположения процесса в таблице процессов. Хэш-метод обеспечивает быстрый доступ к специфическим структурам данных задачи, если ядру известен ее PID. Каждая задача таблицы процессов представляется в виде структуры `task_struct`, служащей в роли дескриптора процесса (т.е. блока управления процессором (PCB)). В структуре `task_struct` хранятся переменные и вложенные структуры, описывающие процесс.

3. Назовите классы потоков в ОС Ubuntu

- 1) Потоки реального времени, обслуживаемые по алгоритму FIFO;
- 2) Потоки реального времени, обслуживаемые в порядке циклической

очереди;

3) Потоки разделения времени.

4. Как используется приоритет планирования при запуске задачи

Планировщик использует приоритет и квант следующим образом. Сначала он вычисляет называемую в системе Linux «добродетелью» (goodness) величину каждого готового процесса по следующему алгоритму:

```
if (class == real_time) goodness = 1000 + priority;
```

```
if (class == timesharing & quantum > 0) goodness = quantum + priority;
```

```
if (class == timesharing && quantum == 0) goodness = 0;
```

Когда нужно принять решение, выбирается поток с максимальным значением «добродетели». Во время работы процесса его квант (переменная quantum) уменьшается на единицу на каждом тике. Центральный процессор отнимается у потока при выполнении одного из следующих условий:

1. Квант потока уменьшился до 0.

2. Поток блокируется на операции ввода-вывода и т.д.

3. В состояние готовности перешел ранее заблокированный поток более высокой «добродетелью».

5. Как можно изменить приоритет для выполняющейся задачи?

С помощью команды `renice -n`