

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

Осипов А.А.

Группа АИ-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

Оглавление

Цель работы	3
Задание кафедры.....	4
Ход работы	5
Вывод	10
Контрольные вопросы.....	11

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.

Команда `echo` - это не системная утилита, у нее нет исполняемого файла. Она существует только внутри интерпретатора Bash.

Командная оболочка `shell` обеспечивает взаимодействие между пользователем и средой операционной системы Linux.

Выведем информацию о текущем интерпретаторе команд с помощью команды `echo $SHELL`:

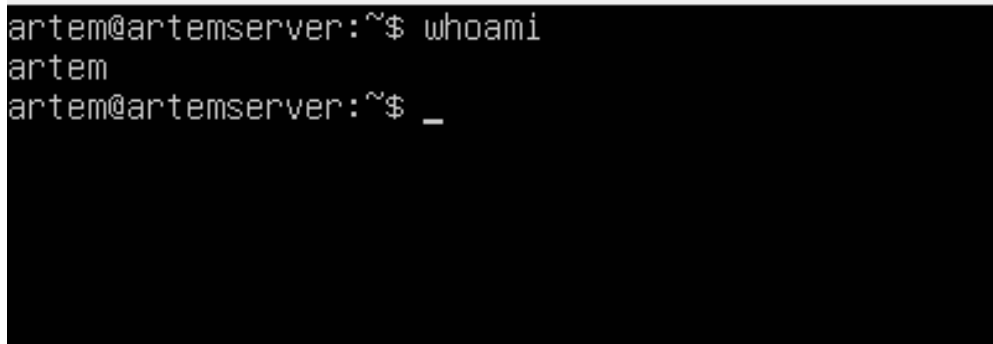


```
artem@artemserver:~$ echo $SHELL
/bin/bash
artem@artemserver:~$
```

Рисунок 1 – Информация о текущем интерпретаторе команд

Команда `whoami` - отображает имя вошедшего в систему пользователя.

Выведем информацию о текущем пользователе с помощью команды `whoami`:

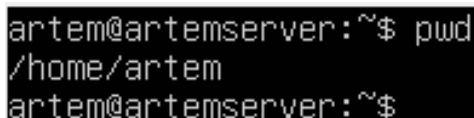


```
artem@artemserver:~$ whoami
artem
artem@artemserver:~$ _
```

Рисунок 2 - Информацию о текущем пользователе

Команда `pwd` - это простая утилита, которая позволяет вывести в терминал путь к текущей папке.

Выведем информацию о текущем каталоге с помощью команды `pwd`:

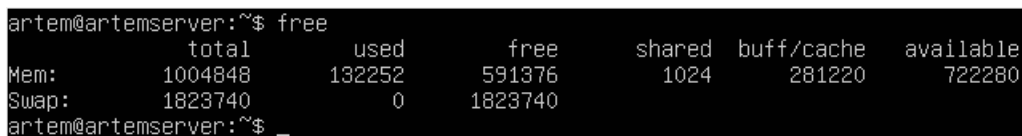


```
artem@artemserver:~$ pwd
/home/artem
artem@artemserver:~$
```

Рисунок 3 - Информация о текущем каталоге

Команда `free` – это простая утилита, которая позволяет легко находить результаты в реальном времени для различных вариантов использования.

Выведем информацию об оперативной памяти и области подкачки:



```
artem@artemserver:~$ free
              total        used        free      shared  buff/cache   available
Mem:           1004848      132252        591376         1024       281220       722280
Swap:          1823740           0        1823740
```

Рисунок 4 - Информацию об оперативной памяти и области подкачки

Total - эта цифра представляет всю существующую память.

Used - вычисление общего значения оперативной памяти системы за вычетом выделенной свободной, разделяемой, буферной и кэш-памяти.

Free – это память, которая не используется ни для каких целей.

Shared, Buffer, и Cache - идентифицируют память, используемую для нужд ядра / операционной системы. Буфер и кеш складываются вместе, а сумма указывается в разделе «buff/cache».

Available - память появляется в более новых версиях `free` и предназначена для того, чтобы дать конечному пользователю оценку того, сколько ресурсов памяти все еще открыто для использования.

Выведем информацию о дисковой памяти:

```
artem@artemserver:~$ df
Filesystem            1K-blocks    Used Available Use% Mounted on
udev                  458724      0    458724   0% /dev
tmpfs                 100488     1024     99464   2% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 9219412 4787856 3943520  55% /
tmpfs                 502424      0    502424   0% /dev/shm
tmpfs                  5120        0      5120   0% /run/lock
tmpfs                 502424      0    502424   0% /sys/fs/cgroup
/dev/sda2             999320    201144    729364  22% /boot
/dev/loop0            56320     56320      0 100% /snap/core18/1880
/dev/loop1            56704     56704      0 100% /snap/core18/1932
/dev/loop2            72320     72320      0 100% /snap/lxd/16922
/dev/loop3            69376     69376      0 100% /snap/lxd/18150
/dev/loop4            30720     30720      0 100% /snap/snapd/8542
/dev/loop5            31744     31744      0 100% /snap/snapd/9721
tmpfs                 100484      0    100484   0% /run/user/1000
artem@artemserver:~$ _
```

Рисунок 5 – Вывод информации о дисковой памяти

Теперь перейдем к выполнению заданий из раздела вывода информации о процессах.

Получим идентификатор текущего процесса (PID):

```
artem@artemserver:~$ echo $$
882
artem@artemserver:~$
```

Рисунок 6 – Идентификатор текущего процесса

Получим идентификатор родительского процесса (PPID):

```
artem@artemserver:~$ echo $PPID
642
artem@artemserver:~$ _
```

Рисунок 7 – Идентификатор родительского процесса

Далее получим идентификатор процесса инициализации системы с помощью команды `pidof`:

```
artem@artemserver:~$ pidof bash
882
artem@artemserver:~$ _
```

Рисунок 8 - идентификатор процесса инициализации системы

Теперь получить информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд. Для этого нам нужно использовать команду `ps` без аргументов:

```
artem@artemserver:~$ ps
  PID TTY          TIME CMD
  882 tty1      00:00:00 bash
  997 tty1      00:00:00 ps
artem@artemserver:~$
```

Рисунок 9 - Информацию о выполняющихся процессах текущего пользователя

Далее отобразим все процессы с помощью `ps -e`:

```
491 ?      00:00:00 kmpath_rdadcd
492 ?      00:00:00 kmpathd
493 ?      00:00:00 kmpath_handlerd
494 ?      00:00:00 multipathd
504 ?      00:00:00 loop0
505 ?      00:00:00 jbd2/sda2-8
506 ?      00:00:00 ext4-rsv-conver
508 ?      00:00:00 loop1
511 ?      00:00:00 loop2
513 ?      00:00:00 loop3
514 ?      00:00:00 loop4
515 ?      00:00:00 loop5
530 ?      00:00:00 systemd-timesyn
568 ?      00:00:00 systemd-network
570 ?      00:00:00 systemd-resolve
584 ?      00:00:00 accounts-daemon
591 ?      00:00:00 cron
592 ?      00:00:00 dbus-daemon
602 ?      00:00:00 networkd-dispat
603 ?      00:00:00 rsyslogd
605 ?      00:00:01 snapd
609 ?      00:00:00 systemd-logind
613 ?      00:00:00 atd
642 tty1    00:00:00 login
646 ?      00:00:00 unattended-upgr
647 ?      00:00:00 polkitd
876 ?      00:00:00 systemd
877 ?      00:00:00 (sd-pam)
882 tty1    00:00:00 bash
925 ?      00:00:03 kworker/0:1-events
967 ?      00:00:00 fwupd
976 ?      00:00:00 kworker/0:2
983 ?      00:00:00 kworker/u2:2-events_power_efficient
995 ?      00:00:00 kworker/u2:1-events_power_efficient
999 ?      00:00:00 kworker/u2:0-events_unbound
1001 tty1   00:00:00 ps
artem@artemserver:~$
```

Рисунок 10 – Все процессы

В последнем разделе нам нужно познакомиться с командами управления процессами.

Выполним следующие действия: выведем информацию о выполняющихся процессах текущего пользователя в текущем интерпретаторе команд(`ps`) и определим текущее значение `nice`. Затем запустим новый экземпляр интерпретатора `bash` с понижением приоритета и узнаем его идентификатор. Затем установим приоритет запущенного интерпретатора равный 6 и получим информацию о процессах `bash`:

```
artem@artemserver:~$ ps
  PID TTY          TIME CMD
   882 tty1      00:00:00 bash
  1005 tty1      00:00:00 ps
artem@artemserver:~$ nice
0
artem@artemserver:~$ nice -n 10 bash
artem@artemserver:~$ echo $$
1007
artem@artemserver:~$ sudo renice -n 6 1007
[sudo] password for artem:
1007 (process ID) old priority 10, new priority 6
artem@artemserver:~$ ps lax | grep bash
 4 1000    882      642  20   0  8264  5132 do_wai S      tty1      0:00 -bash
 0 1000    1007     882  26   6  8272  5128 do_wai SN    tty1      0:00 bash
 0 1000    1017     1007  26   6  6300   736 -      RN+    tty1      0:00 grep --color=auto bash
artem@artemserver:~$
```

Рисунок 11 – Операции с созданным интерпретатором

Вывод

В ходе выполнения лабораторной работы я ознакомился 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: квант потока уменьшился до 0, поток блокируется на операции ввода-вывода и т.д., в состоянии готовности перешел ранее заблокированный поток более высокой «добродетелью».

5. Как можно изменить приоритет для выполняющейся задачи?

С помощью команды `renice -n`