Липецкий государственный технический университет

Кафедра прикладной математики

Отчет по лабораторной работе №4
«Управление процессами ОС Ubuntu.»

Студент _____ Комолых Т.О.
подпись, дата фамилия, инициалы

Группа ПМ-18

Руководитель
доц., к.п.н. кафедры АСУ _____ Кургасов В. В.
ученая степень, ученое звание подпись, дата фамилия, инициалы

Липецк 2020 г.

Содержание

Цель работы

Ознакомиться со средствами управления процессами ОС Ubuntu.

Задание кафедры

Вывод общей информации о системе:

1) Вывод информации о текущем интерпретаторе команд.
2) Вывод информации о текущем пользователе.
3) Вывод информации о текущем каталоге.
4) Вывод информации об оперативной памяти и области подкачки.
5) Вывод информации о дисковой памяти.

Команды получения информации о процессах:

1) Определение идентификатора текущего процесса (PID).
2) Определение идентификатора родительского процесса (PPID).
3) Определение идентификатора процесса инициализации системы.
4) Получение информации о выполняющихся процессах текущего прользователя в текущем интерпретаторе команд.
5) Вывод всех процессов.

Команды управления процессами:

1) Получение информации о выполняющихся процессах текущего пользователя в текущем интерпретаторе.
2) Определение текущего значения nice по умолчанию.
3) Запуск интерпретатора bash с понижением приоритета: nice -n 10 bash
4) Определение PID запущенного интерпретатора.
5) Изменение приоритета запущенного интерпретатора на 5: renice -n 5 <PID процесса>
6) Получение информации о процессах bash: ps lax | grep bash

Выполнение работы

Вывод общей информации о системе:

Вывод информации о текущем интерпретаторе команд.

В большинстве операционных систем Linux, bash является принятым по умолчанию интерпретатором командной строки. Для определения текущего интерпретатора команд (рисунок 1) используется команда echo $SHELL.

```
tatyana@arianrod:~$ echo $SHELL
/bin/bash
```

Рисунок 1.

Вывод информации о текущем пользователе.

Для получения информации о текущем пользователе (рисунок 2) используется команда whoami.

```
tatyana@arianrod:~$ whoami
tatyana
```

Рисунок 2.

Вывод информации о текущем каталоге.

Для получения информации о текущем каталоге (рисунок 3) используется команда pwd.

```
tatyana@arianrod:~$ pwd
/home/tatyana
```

Рисунок 3.

Вывод информации об оперативной памяти и области подкачки.

Команда free выводит информацию (рисунок 4) об общем количестве оперативной памяти, о количестве занятой и свободной памяти, а также об использовании файла подкачки. По умолчанию объём памяти выводится в килобайтах.

```
tatyana@arianrod:~$ free
               total        used        free      shared  buff/cache   available
Mem:         1004848      155888      316076        1044      532884      691996
Swap:        1235964           0     1235964
```

Рисунок 4.

Вывод информации о дисковой памяти.

Команда df отображает информацию (рисунок 5) об имени устройства, общем количестве блоков, используемом дисковом пространстве, общем дисковом пространстве, доступном дисковом пространстве и точках монтирования в файловой системе.

```
tatyana@arianrod:~$ df
Filesystem                         1K-blocks    Used Available Use% Mounted on
udev                                  458724       0    458724   0% /dev
tmpfs                                 100488    1044     99444   2% /run
/dev/mapper/ubuntu--vg-ubuntu--lv    7155192 5045852   1726160  75% /
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  296884    633624  32% /boot
/dev/loop0                            100096  100096         0 100% /snap/core/10185
/dev/loop2                             56704   56704         0 100% /snap/core18/1885
/dev/loop1                            100096  100096         0 100% /snap/core/10126
/dev/loop3                             56704   56704         0 100% /snap/core18/1932
/dev/loop5                             72320   72320         0 100% /snap/lxd/16922
/dev/loop6                             31744   31744         0 100% /snap/snapd/9607
/dev/loop7                             31744   31744         0 100% /snap/snapd/9721
/dev/loop8                              4352    4352         0 100% /snap/tree/18
/dev/loop9                             69376   69376         0 100% /snap/lxd/18150
tmpfs                                 100484       0    100484   0% /run/user/1000
```

Рисунок 5.

Команды получения информации о процессах:

Определение идентификатора текущего процесса (PID).

Для определения идентификатора текущего процесса (рисунок 6) используется команда echo $$, где $$ - PID текущего процесса.

```
tatyana@arianrod:~$ echo $$
1636
```

Рисунок 6.

Определение идентификатора родительского процесса (PPID).

Для определения идентификатора родительского процесса (рисунок 7) используется команда echo $PPID.

```
tatyana@arianrod:~$ echo $PPID
1540
```

Рисунок 7.

Определение идентификатора процесса инициализации системы.

При вызове без какой-либо опции pidof будет печатать PID всех запущенных программ, которые совпадают с заданным именем. Для вывода идентификатора процесса инициализации системы (рисунок 8) используется команда pidof init. Результатом данной команды является PID, равный 1.Вывод считается верным, так как первым всегда запускается процесс init с номером 1, который порождает все остальные процессы.

```
tatyana@arianrod:~$ pidof init
1
```

Рисунок 8.

Получение информации о выполняющихся процессах текущего прользователя в текущем интерпретаторе команд.

Для просмотра процессов в текущей оболочке (рисунок 9) используется команда терминала ps, выводящаяя идентификатор процесса, имя управляющего терминала, количество времени центрального процессора, которое потребил процесс, и имя команды.

```
tatyana@arianrod:~$ ps
    PID TTY          TIME CMD
   1636 tty1     00:00:00 bash
   1659 tty1     00:00:00 ps
```

Рисунок 9.

Вывод всех процессов.

Добавление опции -e (выбрать все процессы) сделает так, что ps перечислит процессы, которые были запущены всеми пользователями, а не только пользователем, который запускает команду ps (рисунок 10). Поскольку это будет длинный список, можно добавить команду less.

Вопросительный знак в столбце TTY означает, что процесс запускался не из окна терминала.

```
tatyana@arianrod:~$ ps -e | less_
```

Рисунок 10.

Перечень всех процессов представлен на рисунках 11, 12 и 13.

```
    PID TTY          TIME CMD
      1 ?        00:00:01 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      5 ?        00:00:00 kworker/0:0-events
      6 ?        00:00:00 kworker/0:0H-kblockd
      9 ?        00:00:00 mm_percpu_wq
     10 ?        00:00:00 ksoftirqd/0
     11 ?        00:00:00 rcu_sched
     12 ?        00:00:00 migration/0
     13 ?        00:00:00 idle_inject/0
     14 ?        00:00:00 cpuhp/0
     15 ?        00:00:00 kdevtmpfs
     16 ?        00:00:00 netns
     17 ?        00:00:00 rcu_tasks_kthre
     18 ?        00:00:00 kauditd
     19 ?        00:00:00 khungtaskd
     20 ?        00:00:00 oom_reaper
     21 ?        00:00:00 writeback
     22 ?        00:00:00 kcompactd0
     23 ?        00:00:00 ksmd
     24 ?        00:00:00 khugepaged
     70 ?        00:00:00 kintegrityd
     71 ?        00:00:00 kblockd
     72 ?        00:00:00 blkcg_punt_bio
     73 ?        00:00:00 tpm_dev_wq
     74 ?        00:00:00 ata_sff
     75 ?        00:00:00 md
     76 ?        00:00:00 edac-poller
     77 ?        00:00:00 devfreq_wq
     78 ?        00:00:00 watchdogd
     81 ?        00:00:00 kswapd0
     82 ?        00:00:00 ecryptfs-kthrea
     84 ?        00:00:00 kthrotld
     85 ?        00:00:00 acpi_thermal_pm
```

Рисунок 11.

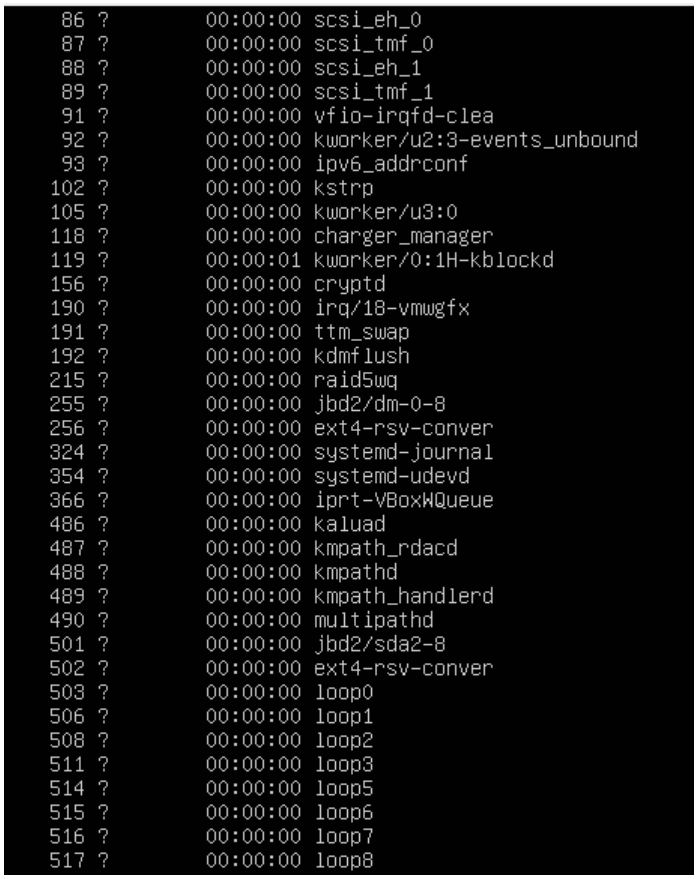

Рисунок 12.

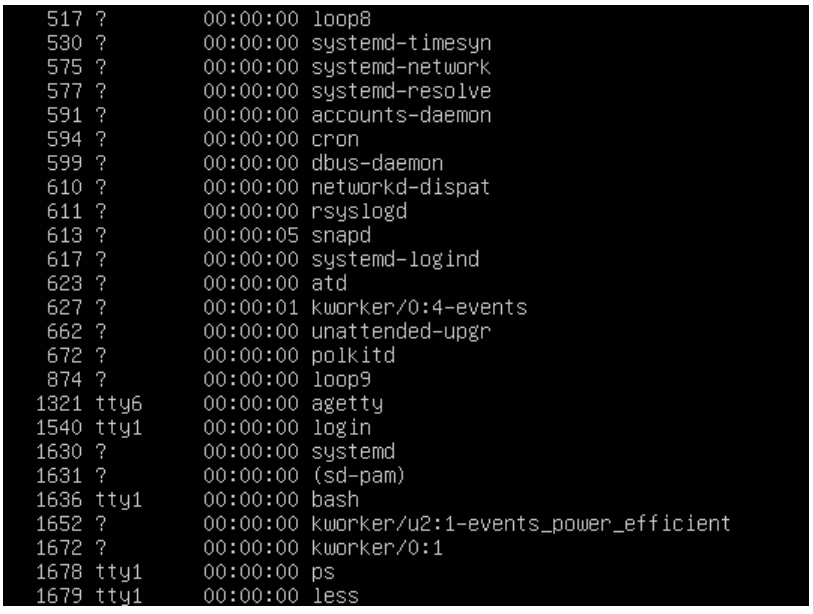

Рисунок 13.

Команды управления процессами:

Получение информации о выполняющихся процессах текущего пользователя в текущем интерпретаторе.

Чтобы вывести больше информации о процессах, используется ps -f (рисунок 14), выводящий

имя пользователя, идентификатор процесса, идентификатор родительского процесса, процент времени CPU, используемого процессом, время запуска процесса, терминал, из которого запущен процесс, общее время процессора и команда запуска процессора.

```
tatyana@arianrod:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
tatyana     1636    1540  0 13:38 tty1     00:00:00 -bash
tatyana     1682    1636  0 13:54 tty1     00:00:00 ps -f
```

Рисунок 14.

Определение текущего значения nice по умолчанию.

Для определения текущего значения приоритета процесса (рисунок 15) используетя команда nice.

```
tatyana@arianrod:~$ nice
0
```

Рисунок 15.

Запуск интерпретатора bash с понижением приоритета и определение PID запущенного интерпретатора.

Чтобы запустить процесс со значением nice, равным 10, можно использовать ключ -n (рисунок 16). Чтобы установить значение nice ниже нуля, требуются права суперпользователя. В противном случае будет установлено значение 0. Поэтому, чтобы задать значение nice меньше 0, необходимо запускать программу как root, или использовать sudo.

```
tatyana@arianrod:~$ nice -n 10 bash
```

Рисунок 16.

В результате команда bash имеет приоритет 10. PID данного процесса равен 1710.

```
    PID USER      PRI  NI  VIRT   RES   SHR S  CPU%  MEM%   TIME+  Command
   1710 tatyana    30  10  7036  4964  3336 S   0.0   0.5   0:00.01 bash
```

Рисунок 17.

Изменение приоритета запущенного интерпретатора на 5.

С помощью команды renice можно изменить значение nice у запущенной программы (рисунок 18). Для этого используется команда renice -n 5 1710. Чтобы изменить значение nice, требуются права суперпользователя. Поэтому, чтобы изменить значение nice, необходимо запускать программу как root, или использовать sudo.

```
tatyana@arianrod:~$ renice -n 5 1710
renice: failed to set priority for 1710 (process ID): Permission denied
tatyana@arianrod:~$ sudo renice -n 5 1710
[sudo] password for tatyana:
```

Рисунок 18.

На рисунке 19 показано, что процесс с PID 1710 изменил свой приоритет с 10 на 5.

```
1710 (process ID) old priority 10, new priority 5
```

Рисунок 19.

Получение информации о процессах bash.

Для отображения информации о значениях nice процессах (рисунок 20) используется команда ps lax. Если нужно найти конкретную папку или файл, то можно передать вывод команды ps в grep через вертикальную черту (|), а уже grep-у параметром передать нужное слово.

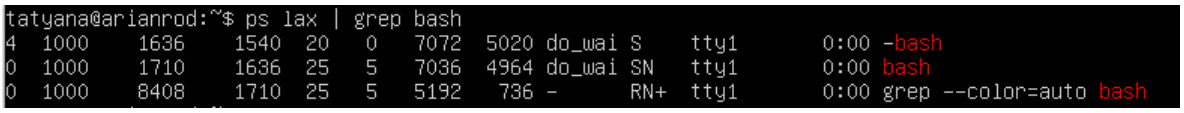

Рисунок 20.

Контрольные вопросы.

Перечислите состояния задачи в ОС Ubuntu.

Сведения о текущем состоянии задачи содержатся в переменной state.

1) Задача переходит в состояние running (выполнения) после выделения ей процессора.
2) При блокировке задача переходит в состояние sleeping (спячки).
3) При остановке работы переходит в состояние stopped (остановки).
4) Состояние zombie (зомби) показывает, что выполнение задачи прекратилось, однако она ещё не была удалена из системы.
5) Задача в состоянии dead (смерти) может быть удалена из системы.
6) Состояния active (активный) и expired (неактивный) используются при планировании выполнения процесса, и поэтому они не сохраняются в переменной state.

Как создаются задачи в ОС Ubuntu?

Задачи создаются путём вызова системной функции clone. Любые обращения к fork и vfork преобразуются в системные вызовы clone во время компиляции.

Функция fork создаёт дочернюю задачу, виртуальная память для которой выделяется по принципу копирования призаписи. Когда дочерний или родительский процесс пытается выполнить запись в страницу памяти, записывющая программа создаёт собственную копию страницы в памяти. Копирование при записи может привести к снижению быстродействия в том случае, когда процесс использует процедуру execve для загрузки новой программы сразу после fork.

Процедура vfork приостанавливает работу родительского процесса в том случае, когда дочерний процесс вызывает функции execve или exit, чтобы обеспечить загрузку дочерним процессам новых страниц до того, как родительский процесс начнёт выполнять бесполезные операции копирования при записи. Процедура vfork позволяет ещё больше повысить производительность благодаря тому, что при её вызове таблицы страниц родительского процесса не копируются в дочерний, поскольку новые таблицы страниц создаются тогда, когда дочерний процесс вызывает функцию execve.

Назовите классы потоков ОС Ubuntu.

В ОС Linux различают три класса потоков:

1) Потоки реального времени, обслуживаемые по алгоритму FIFO.
2) Потоки реального времени, обслуживаемые в порядке циклической очереди.
3) Потоки разделения времени.

Как используется приоритет планирования при запуске задачи?

У каждого потока есть приоритет планирования. Значение по умолчанию равно 20, но оно может быть изменено при помощи системного вызова nice(value), вычитающего значение value из 20. Поскольку value должно находиться в диапазоне от -20 до +19, приоритеты всегда попадают в промежуток от 1 до 40.

Цель алгоритма планирования состоит в том, чтобы обеспечить грубое пропорциональное соответствие качества обслуживания приоритету, то есть чем выше приоритет, тем меньше должно быть время отклика и тем большая доля процессорного времени достаётся процессу.

Как можно изменить приоритет для выполняющейся задачи?

Изменить приоритет выполняющейся задачи позволяет команда renice со значением PID процесса. Например: renice -n 10 1710

Согласно правилам, обычный пользователь может только увеличивать значение nice (уменьшать приоритет) любого процесса. Если попробовать изменить значение nice с 15 до 10, появится сообщение об ошибке.

Также, команда renice позволяет суперпользователю изменять значение nice процессов любого пользователя. Это делается с помощью ключа -u.

Вывод

В ходе лабораторной работы были усвоены знания по работе со средствами управления процессами ОС Ubuntu.

Список литературы

[1] Львовский, С.М. Набор и верстка в системе LaTeX [Текст] / С.М. Львовский. М.: МЦНМО, 2006. — 448 с.