



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

О Т Ч Е Т

по лабораторной работе № 4

Название: Исследование способов организации оперативной памяти и взаимодействия процессов

Дисциплина: Операционные системы

Студент

ИУ6-53Б
(Группа)

(Подпись, дата)

В.К. Залыгин
(И.О. Фамилия)

Преподаватель

(Подпись, дата)

В.Ю. Мельников
А.М. Суровов
(И.О. Фамилия)

Москва, 2024

ВВЕДЕНИЕ

Цель лабораторной работы

Цель данной работы — получение теоретических и практических сведений об управлении процессами, потоками и оперативной памятью в UNIX-подобных системах и в Linux в частности.

Выполнение

- Открыть в текстовом браузере некую страницу и перевести его в фоновый режим

Сначала откроем в консольном браузере сайт и уведем его в фон.

```
[21]+ Stopped w3m google.com
[user@zalugin ~]$
```

Рисунок 1 – Результат перевода браузера в фоновый режим

- Запустить ещё два экземпляра текстового браузера в фоновом режиме

Откроем еще две страницы и уведем их в фон.

```
1747 tty1 T 0:00 w3m google.com
1754 tty1 T 0:00 w3m ya.ru
1755 tty1 T 0:00 w3m yandex.ru
```

Рисунок 2 – Еще два запущенных в фоне процесса

- Найти процесс, максимально нагружающий процессор

Используем утилиту top и посмотрим самый ресурсозатратный процесс.

```
top - 12:19:43 up 1:21, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 80 total, 1 running, 40 sleeping, 3 stopped, 0 zombie
%Cpu(s): 0.0 us, 1.3 sy, 0.0 ni, 98.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem: 1003252 total, 158220 used, 845032 free, 25716 buffers
MiB Swap: 1046524 total, 0 used, 1046524 free, 60512 cached
```

PID	USER	PR	NI	UIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1759	user	20	0	13064	2660	2228	R	0.660	0.265	0:00.96	top
1745	root	20	0	0	0	0	I	0.330	0.000	0:00.32	kworker/0+
1	root	20	0	10816	1556	1420	S	0.000	0.155	0:01.00	init
2	root	20	0	0	0	0	S	0.000	0.000	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.000	0.000	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.000	0.000	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0.000	0.000	0:00.00	kworker/0+
7	root	20	0	0	0	0	I	0.000	0.000	0:00.00	kworker/u+
8	root	0	-20	0	0	0	I	0.000	0.000	0:00.00	mm_percpu+
9	root	20	0	0	0	0	S	0.000	0.000	0:00.44	ksoftirqd+
10	root	20	0	0	0	0	I	0.000	0.000	0:00.03	rcu_preem+
11	root	20	0	0	0	0	I	0.000	0.000	0:00.00	rcu_sched
12	root	20	0	0	0	0	I	0.000	0.000	0:00.00	rcu_bh
13	root	rt	0	0	0	0	S	0.000	0.000	0:00.19	migration+
15	root	20	0	0	0	0	S	0.000	0.000	0:00.00	cpuhp/0
16	root	20	0	0	0	0	S	0.000	0.000	0:00.00	kdevtmpfs
17	root	0	-20	0	0	0	I	0.000	0.000	0:00.00	netns
18	root	20	0	0	0	0	S	0.000	0.000	0:00.00	rcu_tasks+

Рисунок 3 – Больше всего ресурсов CPU потребляет программа top

- «Убить» первый процесс браузера в котором открыта 1 страница

С помощью команды ps а посмотрим все процессы и найдем процессы браузеров. После чего завершим один из них принудительно.

```

user@zalygin ~]$ ps a | grep w3m
1747 tty1      T      0:00 w3m google.com
1754 tty1      T      0:00 w3m ya.ru
1755 tty1      T      0:00 w3m yandex.ru
1766 tty1      S+     0:00 grep --color=auto w3m
user@zalygin ~]$ kill 1747
user@zalygin ~]$ ps a | grep w3m
1747 tty1      T      0:00 w3m google.com
1754 tty1      T      0:00 w3m ya.ru
1755 tty1      T      0:00 w3m yandex.ru
1768 tty1      S+     0:00 grep --color=auto w3m
user@zalygin ~]$ kill -9 1747
[2]  ■■■■■ w3m google.com
user@zalygin ~]$ ps a | grep w3m
1754 tty1      T      0:00 w3m ya.ru
1755 tty1      T      0:00 w3m yandex.ru
1770 tty1      S+     0:00 grep --color=auto w3m
user@zalygin ~]$

```

Рисунок 4 – Принудительное завершение процесса первого браузера

- Вывести список всех процессов всех пользователей

Вывод всех процессов пользователей командой ps a.

```

root      1108  0.0  0.8 430640 8832 ?        Ssl  10:58  0:00 /usr/sbin/Modem
polkitd   1140  0.0  1.6 529588 16528 ?        Sl   10:58  0:00 /usr/libexec/po
root      1179  0.0  1.3 373128 13120 ?        Ssl  10:58  0:00 /usr/sbin/Netwo
root      1228  0.0  0.0  0 0 ?        I<   10:58  0:00 lip6_addrconfl
root      1234  0.0  0.3 45340 3788 ?        Ss   10:58  0:00 /usr/libexec/bl
syslogd   1369  0.0  0.1 12644 1572 ?        Ss   10:58  0:00 /sbin/syslogd -
klogd     1420  0.0  0.1 6340 1584 ?        Ss   10:58  0:00 /sbin/klogd -c
root      1525  0.0  0.1 19320 1956 ?        Sns  10:58  0:00 /usr/sbin/crond
avahi     1579  0.0  0.3 51772 3536 ?        S    10:58  0:00 avahi-daemon: r
avahi     1580  0.0  0.0 51644 368 ?        S    10:58  0:00 avahi-daemon: c
root      1631  0.0  0.0 49076 608 ?        Ss   10:58  0:00 /usr/sbin/sshd
root      1643  0.0  0.3 72292 3500 tty1     Ss   10:58  0:00 /bin/login
root      1644  0.0  0.0 4200 636 tty2     Ss+  10:58  0:00 /sbin/mingetty
root      1645  0.0  0.0 4200 736 tty3     Ss+  10:58  0:00 /sbin/mingetty
root      1646  0.0  0.0 4200 740 tty4     Ss+  10:58  0:00 /sbin/mingetty
root      1647  0.0  0.0 4200 640 tty5     Ss+  10:58  0:00 /sbin/mingetty
root      1648  0.0  0.0 4200 680 tty6     Ss+  10:58  0:00 /sbin/mingetty
user      1669  0.0  0.3 15716 3444 tty1     S    10:58  0:01 -bash
root      1745  0.1  0.0  0 0 ?        I    12:15  0:00 [kworker/0:1-ev
user      1754  0.0  0.5 50472 5720 tty1     T    12:17  0:00 w3m ya.ru
user      1755  0.0  0.5 50472 5684 tty1     T    12:18  0:00 w3m yandex.ru
root      1762  0.1  0.0  0 0 ?        I    12:20  0:00 [kworker/0:2-at
root      1771  0.1  0.0  0 0 ?        I    12:25  0:00 [kworker/0:0-ev
user      1772  0.0  0.2 10816 2420 tty1     R+   12:26  0:00 ps aux
user@zalygin ~]$

```

Рисунок 5 – Вывод всех процессов всех пользователей

- Просмотреть список процессов постранично

Так как список выше, чем размер терминала, удобно будет использовать команду последовательной выдачи строк more.

```

USER      PID %CPU %MEM    USZ    RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  10816  1556 ?        Ss   10:58   0:01 init [4]
root         2  0.0  0.0      0      0 ?        S    10:58   0:00 [kthreadd]
root         3  0.0  0.0      0      0 ?        I<   10:58   0:00 [rcu_gp]
root         4  0.0  0.0      0      0 ?        I<   10:58   0:00 [rcu_par_gp]
root         6  0.0  0.0      0      0 ?        I<   10:58   0:00 [kworker/0:0H-k
bl
root         7  0.0  0.0      0      0 ?        I    10:58   0:00 [kworker/u2:0-s
cl
root         8  0.0  0.0      0      0 ?        I<   10:58   0:00 [mm_percpu_wq]
root         9  0.0  0.0      0      0 ?        S    10:58   0:00 [ksoftirqd/0]
root        10  0.0  0.0      0      0 ?        I    10:58   0:00 [rcu_preempt]
root        11  0.0  0.0      0      0 ?        I    10:58   0:00 [rcu_sched]
root        12  0.0  0.0      0      0 ?        I    10:58   0:00 [rcu_bh]
root        13  0.0  0.0      0      0 ?        S    10:58   0:00 [migration/0]
root        15  0.0  0.0      0      0 ?        S    10:58   0:00 [cpuhp/0]
root        16  0.0  0.0      0      0 ?        S    10:58   0:00 [kdevtmpfs]
root        17  0.0  0.0      0      0 ?        I<   10:58   0:00 [netns]
root        18  0.0  0.0      0      0 ?        S    10:58   0:00 [rcu_tasks_kthr
el
root        19  0.0  0.0      0      0 ?        S    10:58   0:00 [kauditd]
root        20  0.0  0.0      0      0 ?        S    10:58   0:00 [khungtaskd]
root        21  0.0  0.0      0      0 ?        S    10:58   0:00 [oom_reaper]

```

Рисунок 6 – Использование утилиты more для последовательной выдачи данных

- Отобрать из вывода команды ps строку, соответствующую процессу «dbus-daemon», определить, где лежит её выполняемый файл и с какими параметрами он запущен

Выведем строку, относящуюся к процессу dbus-daemon

```
user@zalygin ~1$ ps aux | grep 'dbus-daemon'
message+ 876  0.0  0.3 49700 3244 ?        Ss   10:58   0:00 /bin/dbus-daemo
n --system
```

Рисунок 7 – Строка вывода ps, соответствующая процессу dbus-daemon

- Записать в файл с именем, содержащим текущее время, строку «-----» и список процессов

Чтобы создать файл с датой и требуемой строкой, можно использовать группу команд.

```
user@zalygin ~1$ export NOW=$(date '+%H-%M-%S') && echo $NOW && touch $NOW && e
cho '-----' >> $NOW && ps >> $NOW
12-42-35
user@zalygin ~1$ cat 12-42-35
-----
  PID TTY          TIME CMD
 1669 tty1        00:00:03 bash
 1754 tty1        00:00:00 w3m
 1755 tty1        00:00:00 w3m
 1840 tty1        00:00:00 ps
```

Рисунок 8 – Создание файла с текущим временем, запись в него строки и списка процессов

- Выполнить команду в фоновом режиме с отсрочкой запуска на 1 минуту. Продемонстрировать, что команда выполнялась именно через минуту.

Чтобы отсрочить команду на минуту, можно ее предварять командой date.

```
user@zalygin ~1$ (sleep 60 && date) & date
[5] 1843
Fri Oct 11 12:43:42 UTC 2024
user@zalygin ~1$ Fri Oct 11 12:44:42 UTC 2024
[5]   Done                    ( sleep 60 && date )
```

Рисунок 9 – Отсрочка команды на 1 минуту

- Отобрать из одного из сформированных файлов строки, относящиеся к одному из процессов.

Используем grep, чтобы вывести строки процессов, относящиеся к браузеру.

```

[user@zalygin ~]$ cat 12-42-35
-----
  PID TTY          TIME CMD
 1669 tty1        00:00:03 bash
 1754 tty1        00:00:00 w3m
 1755 tty1        00:00:00 w3m
 1840 tty1        00:00:00 ps
[user@zalygin ~]$ cat 12-42-35 | grep w3m
 1754 tty1        00:00:00 w3m
 1755 tty1        00:00:00 w3m
[user@zalygin ~]$

```

Рисунок 10 – Фильтрация строк, относящихся только к утилите w3m

- Вывести результаты работы произвольной команды в один файл, а сообщения об ошибках в другой. Продемонстрировать правильность работы.

Чтобы перенаправить потоки вывода и ошибок, нужно использовать специальные операторы >, 2>.

```

[user@zalygin ~]$ (echo "stdout" && echo "stderr" 1>&2 ) >out 2>err
[user@zalygin ~]$ cat out
stdout
[user@zalygin ~]$ cat err
stderr
[user@zalygin ~]$

```

Рисунок 11 – Вывод потока вывода и потока ошибок в отдельные файлы

- Выполнить произвольную команду с ограничением использования процессорного времени 300 секунд и выводом результатов и сообщений об ошибках в один файл.

Для выполнения операции с ограничением по времени работы, можно использовать команду timeout.

```

[root@zalygin ~]# bash -c 'timeout -s 9 300 apt-get update' >out 2>&1
[root@zalygin ~]# cat out
Get:1 http://ftp.altlinux.org p8/branch/x86_64 release [1091B]
Get:2 http://ftp.altlinux.org p8/branch/x86_64-i586 release [537B]
Get:3 http://ftp.altlinux.org p8/branch/noarch release [885B]
Fetched 2513B in 1s (1506B/s)
Hit http://ftp.altlinux.org p8/branch/x86_64/classic pkglist
Hit http://ftp.altlinux.org p8/branch/x86_64/classic release
Hit http://ftp.altlinux.org p8/branch/x86_64-i586/classic pkglist
Hit http://ftp.altlinux.org p8/branch/x86_64-i586/classic release
Hit http://ftp.altlinux.org p8/branch/noarch/classic pkglist
Hit http://ftp.altlinux.org p8/branch/noarch/classic release
Reading Package Lists...
Building Dependency Tree...
[root@zalygin ~]#

```

Рисунок 12 – Выполнение операции с ограничением по времени работы

Создаем задание, пишущее в файл текущую дату ежедневно.

- Настроить cron на выполнение команды ежедневно в заданное время. Продемонстрировать правильность работы.

```

#minute (0-59),
#|      hour (0-23),
#|      |      day of the month (1-31),
#|      |      |      month of the year (1-12),
#|      |      |      |      day of the week (0-6 with 0=Sunday).
#|      |      |      |      |      commands
10 13 * * * date >>/tmp/crondate
~

```

Рисунок 13 – Установка команды с запуском ежедневно в 13:10

После смотрим на содержимое файла.

```

11 000 11 10:10:01 010 0001
[root@zalygin ~]# cat /tmp/crondate
Fri Oct 11 13:10:01 UTC 2024
[root@zalygin ~]# _

```

Рисунок 14 – Результат выполнения задания

ЗАКЛЮЧЕНИЕ

В результате выполнения работы были получены теоретические и практические сведения об управлении процессами, потоками и оперативной памятью в UNIX-подобных системах и в Linux в частности.