

Домашнее задание 2

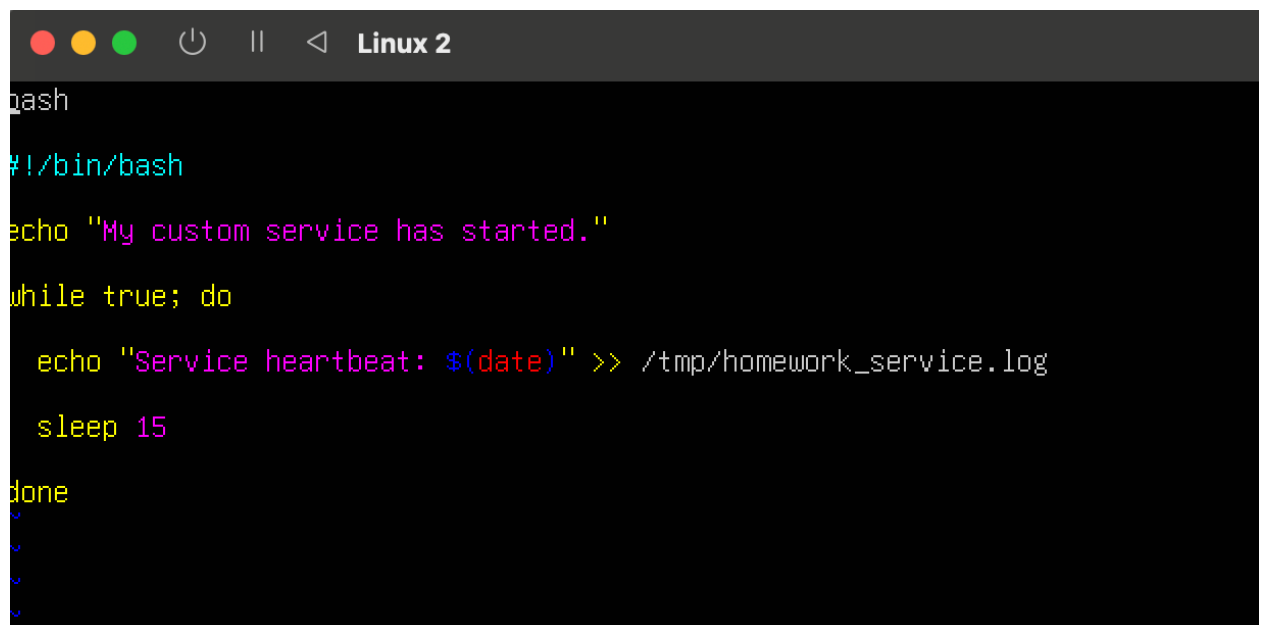
Linux работа с памятью и процессами

Суханов М.Ю.

Задание 1. Systemd (25 баллов)

1. Создайте bash-скрипт /usr/local/bin/homework_service.sh.

```
ubuntu1@serverubuntu:/usr/local/bin$ ls  
homework_service.sh
```



```
#!/bin/bash  
  
echo "My custom service has started."  
  
while true; do  
    echo "Service heartbeat: $(date)" >> /tmp/homework_service.log  
    sleep 15  
done
```

2 / 3. Создайте systemd unit файл для скрипта, который бы переживал любые обновления системы. Убедитесь, что сервис сам перезапускается в случае падения через 15 секунд. / Запустите сервис и убедитесь, что он работает.

Добавим по указанному пути конфигурационный файл сервиса

```
ubuntu1@serverubuntu:/etc/systemd/system$ ls | grep "^home"  
homework_service.service
```

Конфиг состоит из компонентов:

- ExecStart — путь к bash-скрипту;
- Restart=always — настройка перезапуска сервиса (при любом падении);
- RestartSec=15 — настройка ожидания перед перезапуском (15 секунд);
- StartLimitBurst=0 — настройка отключения ограничения на количество рестартов (не позволит systemd заблокировать сервис);
- WantedBy=multi-user.target — настройка автоматического запуска сервиса при старте системы.

```
[Unit]
Description=Homework Service
After=network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/homework_service.sh
Restart=always
RestartSec=15
StartLimitBurst=0

[Install]
WantedBy=multi-user.target
~
~
```

Запустим сервис командами:

```
ubuntu1@serverubuntu:/etc/systemd/system$ sudo systemctl enable homework_service.service
Created symlink '/etc/systemd/system/multi-user.target.wants/homework_service.service' → '/etc/systemd/system/homework_service.service'.
ubuntu1@serverubuntu:/etc/systemd/system$ sudo systemctl start homework_service.service
ubuntu1@serverubuntu:/etc/systemd/system$ sudo systemctl status homework_service.service
homework_service.service - Homework Service
   Loaded: loaded (/etc/systemd/system/homework_service.service; enabled; preset: enabled)
   Active: active (running) since Mon 2025-10-06 17:46:20 UTC; 2min 9s ago
 Invocation: 35063684f0e84809ad0dbfdd04710101
   Main PID: 14553 (homework_service)
     Tasks: 2 (limit: 3932)
    Memory: 628K (peak: 1.6M)
       CPU: 490ms
   CGroup: /system.slice/homework_service.service
           └─14553 /bin/bash /usr/local/bin/homework_service.sh
             14706 sleep 15

окт 06 17:46:20 serverubuntu systemd[1]: homework_service.service: Scheduled restart job, restart counter is at 83.
окт 06 17:46:20 serverubuntu systemd[1]: Started homework_service.service - Homework Service.
окт 06 17:46:20 serverubuntu homework_service.sh[14553]: My custom service has started.
```

Можно увидеть, что сервис запущен под процессом с номером 14553.

Посмотрим логи:

```
ubuntu1@serverubuntu:/tmp$ ls -la
total 8
drwxrwxrwt 12 root root 260 окт 6 17:46 .
drwxr-xr-x 20 root root 4096 сен 30 08:30 ..
drwxrwxrwt 2 root root 40 окт 6 16:38 .font-unix
-rw-r--r-- 1 root root 676 окт 6 17:49 homework_service.log
drwxrwxrwt 2 root root 40 окт 6 16:38 .ICE-unix
drwx----- 2 root root 40 окт 6 16:38 snap-private-tmp
drwxr-xr-x 8 root root 60 окт 6 17:47 snap-private-tmp
```

```

ubuntu1@serverubuntu:/tmp$ cat homework_service.log
Service heartbeat: Пн 06 окт 2025 17:46:20 UTC
Service heartbeat: Пн 06 окт 2025 17:46:35 UTC
Service heartbeat: Пн 06 окт 2025 17:46:50 UTC
Service heartbeat: Пн 06 окт 2025 17:47:05 UTC
Service heartbeat: Пн 06 окт 2025 17:47:20 UTC
Service heartbeat: Пн 06 окт 2025 17:47:35 UTC
Service heartbeat: Пн 06 окт 2025 17:47:50 UTC
Service heartbeat: Пн 06 окт 2025 17:48:05 UTC
Service heartbeat: Пн 06 окт 2025 17:48:20 UTC
Service heartbeat: Пн 06 окт 2025 17:48:35 UTC
Service heartbeat: Пн 06 окт 2025 17:48:50 UTC
Service heartbeat: Пн 06 окт 2025 17:49:05 UTC
Service heartbeat: Пн 06 окт 2025 17:49:20 UTC
Service heartbeat: Пн 06 окт 2025 17:49:35 UTC
ubuntu1@serverubuntu:/tmp$

```

Попробуем убить сервис и посмотрим, что будет:

```

ubuntu1@serverubuntu:/tmp$ sudo kill -9 14553
ubuntu1@serverubuntu:/tmp$ sudo systemctl status homework_service.service
● homework_service.service - Homework Service
   Loaded: loaded (/etc/systemd/system/homework_service.service; enabled; preset: enabled)
   Active: activating (auto-restart) (Result: signal) since Mon 2025-10-06 17:51:03 UTC; 6s ago
  Invocation: 35063684f0e84809ad0dbfdd04710101
    Process: 14553 ExecStart=/usr/local/bin/homework_service.sh (code=killed, signal=KILL)
   Main PID: 14553 (code=killed, signal=KILL)
    Mem peak: 1.6M
       CPU: 1.061s
ubuntu1@serverubuntu:/tmp$ sudo systemctl status homework_service.service
● homework_service.service - Homework Service
   Loaded: loaded (/etc/systemd/system/homework_service.service; enabled; preset: enabled)
   Active: active (running) since Mon 2025-10-06 17:51:18 UTC; 10s ago
  Invocation: 89b78efb9d1546a6bf861b8e8f3357a7
    Main PID: 14758 (homework_service)
      Tasks: 2 (limit: 3932)
    Memory: 588K (peak: 1.6M)
       CPU: 80ms
    CGroup: /system.slice/homework_service.service
            └─14758 /bin/bash /usr/local/bin/homework_service.sh
              └─14760 sleep 15

окт 06 17:51:18 serverubuntu systemd[1]: homework_service.service: Scheduled restart job, restart counter is at 84.
окт 06 17:51:18 serverubuntu systemd[1]: Started homework_service.service - Homework Service.
окт 06 17:51:18 serverubuntu homework_service.sh[14758]: My custom service has started.
ubuntu1@serverubuntu:/tmp$

```

Видно, что сервис сам восстанавливается и продолжает работу.

4. Используя `systemd-analyze`, покажите топ-5 `systemd unit`'ов стартующих дольше всего.

Для этого воспользуемся командой - `systemd-analyze blame`, которая выводит список всех `systemd unit`'ов, отсортированных по времени запуска (от медленного к быстрому).

```

ubuntu1@serverubuntu:/tmp$ systemd-analyze blame | head -n 5
3min 15.972s apt-daily-upgrade.service
33.005s motd-news.service
15.132s homework_service.service
7.325s dev-mapper-ubuntu\x2d\x2dvg\x2dubuntu\x2d\x2dlv.device
6.625s snapd.seeded.service
ubuntu1@serverubuntu:/tmp$

```

Из скриншота видно, что наш сервис входит в топ 3 долго-стартующих сервисов. На первом месте сервис обновления пакетов в фоне, а на втором механизм фонового обновления Debian-Ubuntu.

Задание 2. Межпроцессное взаимодействие (IPC) с разделяемой памятью (20 баллов)

1 / 2. Создайте шареную память. На любом языке программирования создайте программу использующую шареную память / Проанализируйте вывод, пока программа запущена.

```
ubuntu1@serverubuntu:~$ ls
shared shared shm_creator.c
ubuntu1@serverubuntu:~$ gcc sh
shared/ shm_creator.c
ubuntu1@serverubuntu:~$ gcc shm_creator.c -o shm_creator
ubuntu1@serverubuntu:~$ ls
shared shared shm_creator shm_creator.c
ubuntu1@serverubuntu:~$ touch homework_key
ubuntu1@serverubuntu:~$ la
bash_history .bashrc shared .lessht shared shm_creator.c .sudo_as_admin_successful
bash_logout .cache homework_key .profile shm_creator .ssh .viminfo
ubuntu1@serverubuntu:~$ ./shm_creator
shared memory segment created.
ID: 0
Key: 0x41020026
Run 'ipcs -m' to see it. Process will exit in 60 seconds...
```

После компиляции кода на С и создания необходимого файла homework_key запустим программу:

```
ubuntu1@serverubuntu:~$ ./shm_creator
Shared memory segment created.
ID: 0
Key: 0x41020026
Run 'ipcs -m' to see it. Process will exit in 60 seconds...
ipcs -m
^C
ubuntu1@serverubuntu:~$ ./shm_creator &
[1] 14873
ubuntu1@serverubuntu:~$ Shared memory segment created.
ID: 0
Key: 0x41020026
Run 'ipcs -m' to see it. Process will exit in 60 seconds...
^C
ubuntu1@serverubuntu:~$ ./shm_creator &
[2] 14875
ubuntu1@serverubuntu:~$ Shared memory segment created.
ID: 0
Key: 0x41020026
Run 'ipcs -m' to see it. Process will exit in 60 seconds...
ipcs -m

----- Shared Memory Segments -----
key      shmid    owner     perms     bytes     nattch   status
0x41020026 0         ubuntu1   666       1024      0
```

В выводе увидим активный сегмент выделяемой памяти:

key – ключ сегмента;

shmid - уникальный идентификатор сегмента;

owner - имя создателя сегмента (ubuntu1);

perms - права доступа к сегменту (например, 666 — чтение/запись для всех);

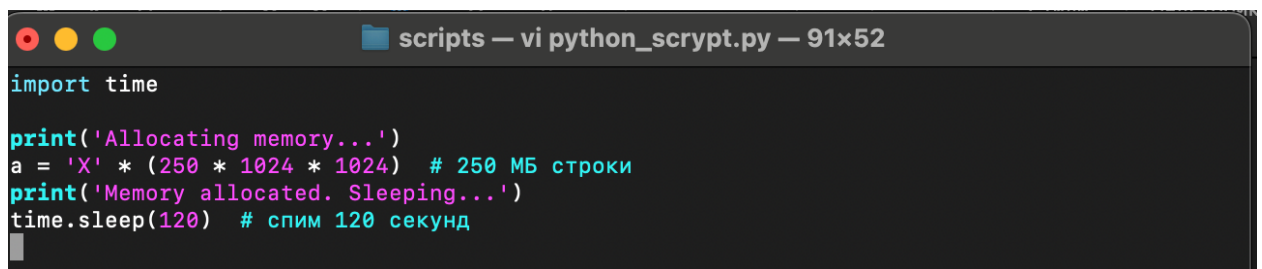
bytes - размер сегмента в байтах (например, 1024 — 1 КБ);

nattch - количество процессов, которые подключены к этому сегменту (0). nattch = 0, потому что процесс создал сегмент, но не присоединился к нему через shmat().

Задание 3. Анализ памяти процессов (VSZ vs RSS) (20 баллов)

1 / 2. Откройте 1 окно терминала и запустите питон скрипт, который запрашивает 250 MiB памяти и держит ее 2 минуты / Объясните почему vsz больше rss, и почему rss далеко не 0.

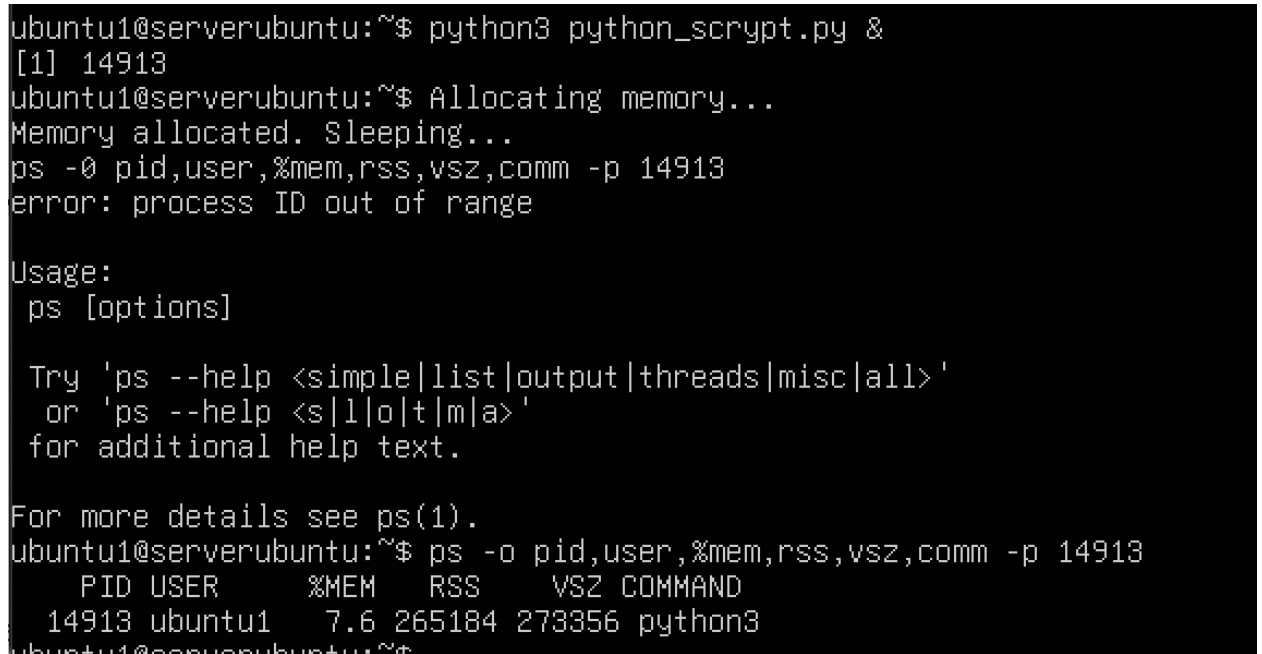
Перепишем скрипт в файл для удобства:



```
scripts — vi python_scrypt.py — 91x52
import time

print('Allocating memory...')
a = 'X' * (250 * 1024 * 1024) # 250 МБ строки
print('Memory allocated. Sleeping...')
time.sleep(120) # спим 120 секунд
```

Запустим Python скрипт в фоне и выполним команду отображения запущенного процесса - "ps -o pid,user,%mem,rss,vsz,comm -p %YOUR_PID%."



```
ubuntu1@serverubuntu:~$ python3 python_scrypt.py &
[1] 14913
ubuntu1@serverubuntu:~$ Allocating memory...
Memory allocated. Sleeping...
ps -o pid,user,%mem,rss,vsz,comm -p 14913
error: process ID out of range

Usage:
ps [options]

Try 'ps --help <simple|list|output|threads|misc|all>'
or 'ps --help <s|l|o|t|m|a>'
for additional help text.

For more details see ps(1).
ubuntu1@serverubuntu:~$ ps -o pid,user,%mem,rss,vsz,comm -p 14913
  PID USER      %MEM    RSS   VSZ COMMAND
  14913 ubuntu1    7.6 265184 273356 python3
ubuntu1@serverubuntu:~$
```

Можно увидеть, что процесс использует 7.6% оперативной памяти. Объем реально используемой оперативной памяти – 265184KB, объем используемой виртуальной памяти, которую процесс запросил – 273356KB. Разница обусловлена тем, что процесс использует не весь объем запрошенной для своей работы памяти. Система выделяет виртуальное пространство заранее, но физическую память (RAM) подключает только тогда, когда процесс реально начинает обращаться к соответствующим областям памяти. Это

называется ленивым выделением памяти. RSS больше нуля, потому что минимально запущенная программа нуждается в базовых ресурсах: в оперативной памяти должны быть размещены исполняемый код, стандартные библиотеки, системные вызовы, стек, и так далее.

Задание 4. NUMA и cgroups (35 баллов)

1. Продемонстрируйте количество NUMA нод на вашем сервере и количество памяти для каждой NUMA ноды

```
ubuntu1@serverubuntu:~$ numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1
node 0 size: 3399 MB
node 0 free: 2000 MB
node distances:
node 0
  0: 10
[1]+  Done                  python3 python_s
```

Через команду `numactl` получили кол-во NUMA нод, она одна. Общий размер ноды – 3400MB, свободно 2000MB.

2. Убедитесь, что вы можете ограничивать работу процессов при помощи `systemd`. / Будет ли работать тест если мы запрашиваем 300M оперативной памяти, а ограничиваем 150M?

`Systemd` позволяет ограничивать ресурсы процессов, в том числе указывать, на каких CPU и памяти они могут работать.

Запустим процесс:

```
sudo systemd-run --unit=highload-stress-test --slice=testing.slice \
--property="MemoryMax=150M" \
--property="CPUWeight=100" \
stress --cpu 1 --vm 1 --vm-bytes 300M --timeout 30s
```

Процесс не запустился:

```

окт 06 19:14:52 serverubuntu systemd[1]: Started highload-stress-test.service - [systemd-run] /usr/bin/stress --cpu 1 --vm 1 --vm-bytes 300M --timeout 30s.
окт 06 19:14:52 serverubuntu stress[15464]: stress: info: [15464] dispatching hogs: 1 cpu, 0 io, 1 vm, 0 hdd
окт 06 19:14:53 serverubuntu systemd[1]: highload-stress-test.service: A process of this unit has been killed by the OOM killer.
окт 06 19:14:53 serverubuntu systemd[1]: highload-stress-test.service: Failed with result 'oom-kill'.
окт 06 19:14:53 serverubuntu systemd[1]: highload-stress-test.service: Consumed 1.961s CPU time, 150M memory peak.
ubuntu1@serverubuntu:~$ sudo systemctl status highload-stress-test
* highload-stress-test.service - [systemd-run] /usr/bin/stress --cpu 1 --vm 1 --vm-bytes 300M --timeout 30s
   Loaded: loaded (/run/systemd/transient/highload-stress-test.service; transient)
   Transient: yes
   Active: failed (Result: oom-kill) since Mon 2025-10-06 19:14:53 UTC; 2min is ago
  Duration: 1.015s
 Invocation: 8d08f3c7e90942a7a896ac7266f4e66e
    Process: 15464 ExecStart=/usr/bin/stress --cpu 1 --vm 1 --vm-bytes 300M --timeout 30s (code=killed, signal=TERM)
   Main PID: 15464 (code=killed, signal=TERM)
  Mem peak: 150M
    CPU: 1.961s

```

Через пропери - "MemoryMax=150M" мы выделяем процессу лишь 150MB памяти, сам же процесс пытается аллоцировать 300MB, отсюда возникает ошибка.

Попробуем уменьшить размер аллоцируемой памяти до 150MB и увеличить предел MemoryMax до 170MB:

```

ubuntu1@serverubuntu:~$ sudo systemctl reset-failed highload-stress-test
ubuntu1@serverubuntu:~$ sudo systemctl run --unit=highload-stress-test --slice=testing.slice --property="MemoryMax=170M" --property="CPUWeight=100" stress --cpu 1
--vm 1 --vm-bytes 150M --timeout 30s &
[1] 15591
ubuntu1@serverubuntu:~$ Running as unit: highload-stress-test.service; invocation ID: c37b6c7e2d9e464d8ca75cedbe1285e9
systemctl status highload-stress-test
* highload-stress-test.service - [systemd-run] /usr/bin/stress --cpu 1 --vm 1 --vm-bytes 150M --timeout 30s
   Loaded: loaded (/run/systemd/transient/highload-stress-test.service; transient)
   Transient: yes
   Active: active (running) since Mon 2025-10-06 19:26:14 UTC; 17s ago
 Invocation: c37b6c7e2d9e464d8ca75cedbe1285e9
    Main PID: 15594 (stress)
      Tasks: 3 (limit: 3932)
     Memory: 150.5M (max: 170M, available: 19.1M, peak: 151.1M)
        CPU: 33.907s
    CGroup: /testing.slice/highload-stress-test.service
            └─15594 /usr/bin/stress --cpu 1 --vm 1 --vm-bytes 150M --timeout 30s
              └─15595 /usr/bin/stress --cpu 1 --vm 1 --vm-bytes 150M --timeout 30s
                └─15596 /usr/bin/stress --cpu 1 --vm 1 --vm-bytes 150M --timeout 30s
окт 06 19:26:14 serverubuntu systemd[1]: Started highload-stress-test.service - [systemd-run] /usr/bin/stress --cpu 1 --vm 1 --vm-bytes 150M --timeout 30s.
окт 06 19:26:14 serverubuntu stress[15594]: stress: info: [15594] dispatching hogs: 1 cpu, 0 io, 1 vm, 0 hdd
[1]+ Done                  sudo systemctl run --unit=highload-stress-test --slice=testing.slice --property="MemoryMax=170M" --property="CPUWeight=100" stress -

```

Можно увидеть, что тест запустился. Просмотрим используемые ресурсы:

```

ubuntu1@serverubuntu:~$ systemctl show highload-stress-test -p MemoryCurrent -p CPUUsageNSec
MemoryCurrent=140918784
CPUUsageNSec=47813604000

```

Процесс потребляет ~ 140MB оперативной памяти (не превышает наш установленный порог в 170MB), общее время использования CPU ~ 48 миллисекунд. Данные параметры позволяют накладывать ограничение на процесс по используемой оперативной памяти и оказываемой нагрузке на процессор.