Министерство науки и высшего образования
Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий
Высшая школа киберфизических систем и управления

ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине «Операционные системы»

Выполнил
студент группы №0902/003 Л.М. Васильев

Проверил
доцент к.т.н. Я.А. Селиверстов

2022

Оглавление

Лабораторная работа №1. OS Linux.

Задание

1) Установить версию Linux Ubuntu (методичка 1 + Видео)

Вывести информацию о версии ОС с помощью команд, указанных в методичке 1:

Установка операционной системы и знакомство с ней
2) Интерфейсы командной строки (методичка 2+ Видео)
1. Создать каталоги students и mentors в домашней директории, а в них — текстовые файлы students_list.txt и mentors_list.txt соответственно.
2. Открыть созданные в п.1 файлы в любом текстовом редакторе и заполнить их (в соответствии с названием) списком Ваших одногруппников и наставников на данном потоке.
3. Переместите файл mentors_list.txt в папку students.
4. Удалите папку mentors.
5. Переименуйте папку students в students_and_mentors.
6. Удалите папку students_and_mentors вместе с содержимым.
7. Подключитесь к машине с Linux по протоколу SSH.
8. Используя дополнительный материал, настроить авторизацию по SSH с использованием ключей.

Решение

```
Last login: Tue Oct  4 21:01:22 on ttys000
lenavasilev@MacBook-Air-Lena ~ % ssh futitisme@192.168.64.3
futitisme@192.168.64.3's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-48-generic aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Вт 04 окт 2022 20:59:48 UTC

  System load:  0.40625          Processes:              128
  Usage of /:   27.3% of 9.75GB   Users logged in:        0
  Memory usage: 5%               IPv4 address for enp0s6: 192.168.64.3
  Swap usage:   0%


41 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
```

```
Last login: Tue Oct  4 18:02:04 2022 from 192.168.64.1
futitisme@futitismeserver:~$ uname -a
Linux futitismeserver 5.15.0-48-generic #54-Ubuntu SMP Fri Aug 26 13:31:33 UTC 2022 aarch64 aarch64 aarch64 GNU/Linux
futitisme@futitismeserver:~$ mkdir /home/futitisme/students
futitisme@futitismeserver:~$ mkdir /home/futitisme/mentors
futitisme@futitismeserver:~$ cd /home/futitisme/students
futitisme@futitismeserver:~/students$ touch students_list.txt
futitisme@futitismeserver:~/students$ ls -1
students_list.txt
futitisme@futitismeserver:~/students$ cd /home/futitisme/mentors
futitisme@futitismeserver:~/mentors$ touch mentors_list.txt
futitisme@futitismeserver:~/mentors$ cd /home/futitisme/students
futitisme@futitismeserver:~/students$ nano students_list.txt
futitisme@futitismeserver:~/students$ cd /home/futitisme/mentors
futitisme@futitismeserver:~/mentors$ nano mentors_list.txt
futitisme@futitismeserver:~/mentors$ mv /home/futitisme/mentors/mentors_list.txt /home/futitisme/students/mentors_list.txt
futitisme@futitismeserver:~/mentors$ ls -1
futitisme@futitismeserver:~/mentors$ cd /home/futitisme/students
futitisme@futitismeserver:~/students$ ls -1
mentors_list.txt
students_list.txt
futitisme@futitismeserver:~/students$ rm -rf /home/futitisme/mentors
futitisme@futitismeserver:~/students$ cd /home/futitisme
futitisme@futitismeserver:~$ ls -1
students
futitisme@futitismeserver:~$ cd /home/futitisme/students
futitisme@futitismeserver:~/students$ mv students students_and_mentors
mv: cannot stat 'students': No such file or directory
futitisme@futitismeserver:~/students$ cd /home/futitisme
futitisme@futitismeserver:~$ mv students students_and_mentors
futitisme@futitismeserver:~$ ls -1
students_and_mentors
futitisme@futitismeserver:~$ rm -rf /home/futitisme/students_and_mentors
futitisme@futitismeserver:~$ ls
futitisme@futitismeserver:~$ ls -1
futitisme@futitismeserver:~$
futitisme@futitismeserver:~$
futitisme@futitismeserver:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/futitisme/.ssh/id_rsa): y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in y
```

```
Your public key has been saved in y.pub
The key fingerprint is:
SHA256:4tyfEqr3HG4sSVGLhorZ2/HRsLvCNtzGqPN5egK89Xs futitisme@futitismeserver
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|        .        |
|     . o .       |
|    . = .        |
| = . ..=S        |
|o = oo+oo        |
|   O Oo*o.       |
|  + X.#oEo .     |
|  .==@oB+.o      |
+----[SHA256]-----+
futitisme@futitismeserver:~$ ssh-copy-id futitisme@192.168.64.3
/usr/bin/ssh-copy-id: ERROR: No identities found
futitisme@futitismeserver:~$ /.ssh/id_rsa.pub | ssh futitisme@192.168.64.3 "mkdir -p ~/.ssh &&
cat >> ~/.ssh/authorized_keys"
-bash: /.ssh/id_rsa.pub: No such file or directory
The authenticity of host '192.168.64.3 (192.168.64.3)' can't be established.
ED25519 key fingerprint is SHA256:v9Ezfu0ANJIXIYrGkAL6ymtOTeuzINWUyIhzP7SowIg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.64.3' (ED25519) to the list of known hosts.
futitisme@192.168.64.3's password:
futitisme@futitismeserver:~$
```

Лабораторная 2. Пользователи. Управление Пользователями и группами

Задание

1. Управление пользователями:

a) создать нового пользователя;

b) убедиться, что информация о нем появилась в соответствующих файлах в системе;

c) удалить созданного пользователя;

`2. Управление группами:

a) создать группу;

b) попрактиковаться в смене групп у пользователей;

c) добавить пользователя в группу, не меняя основной;

d) удалить пользователя из группы.

3. Создать пользователя с правами суперпользователя. Проверить результат.

4. * Используя дополнительные материалы, выдать одному из созданных пользователей право на выполнение ряда команд, требующих прав суперпользователя (команды выбираем на своё усмотрение).

Решение

```
Last login: Tue Oct  4 20:44:57 2022 from 192.168.64.3
1)futitisme@futitismeserver:~$ sudo adduser kira

futitisme@futitismeserver:~$ cat /etc/passwd
...
kira:x:1007:1010:,,,:/home/kira:/bin/bash

futitisme@futitismeserver:~$ whoami
futitisme
futitisme@futitismeserver:~$ sudo deluser kira
Removing user `kira' ...
Warning: group `kira' has no more members.
Done.

2)futitisme@futitismeserver:~$ sudo groupadd group_0
futitisme@futitismeserver:~$ cat /etc/group | grep group_0
group_0:x:1010:
futitisme@futitismeserver:~$ sudo usermod -aG group_0 user_0
usermod: user 'user_0' does not exist
futitisme@futitismeserver:~$ sudo usermod -aG group_0 user123
futitisme@futitismeserver:~$ id user123
```

uid=1006(user123) gid=1009(user123) groups=1009(user123),27(sudo),1010(group_0)

3)futitisme@futitismeserver:~$ sudo usermod -aG sudo user123
futitisme@futitismeserver:~$ su user123
Password:
user123@futitismeserver:/home/futitisme$ sudo ls -la /root
[sudo] password for user123:
total 28
drwx------ 5 root root 4096 окт 4 21:03 .
drwxr-xr-x 19 root root 4096 окт 4 15:43 ..
-rw-r--r-- 1 root root 3106 окт 15 2021 .bashrc
drwxr-xr-x 3 root root 4096 окт 4 21:03 .local
-rw-r--r-- 1 root root 161 июл 9 2019 .profile
drwx------ 3 root root 4096 окт 4 18:56 snap
drwx------ 2 root root 4096 окт 4 18:56 .ssh

4)futitisme@futitismeserver:~$ sudo adduser elena

futitisme@futitismeserver:~$ su elena

elena@futitismeserver:/home/futitisme$ sudo deluser user

[sudo] password for elena:

elena is not in the sudoers file. This incident will be reported.

elena@futitismeserver:/home/futitisme$ exit
exit
futitisme@futitismeserver:~$ sudo visudo
(Elena = ALL=/usr/sbin/deluser)
futitisme@futitismeserver:~$ su elena

elena@futitismeserver:/home/futitisme$ sudo deluser user

Removing user `user' ...
Warning: group `user' has no more members.
userdel: user user is currently used by process 1550
/usr/sbin/deluser: `/sbin/userdel user' returned error code 8. Exiting.

Лабораторная 3. Загрузка ОС и процессы

Задание

1. Потоки ввода/вывода. Создать файл, используя команду echo. Используя команду cat, прочитать содержимое всех файлов каталога etc, ошибки перенаправить в отдельный файл.

2. Конвейер (pipeline). Использовать команду cut на вывод длинного списка каталога, чтобы отобразить только права доступа к файлам. Затем отправить в конвейере этот вывод на sort и uniq, чтобы отфильтровать все повторяющиеся строки.

3. Управление процессами.) Изменить конфигурационный файл службы SSH: /etc/ssh/sshd_config, отключив аутентификацию по паролю PasswordAuthentication no. Выполните рестарт службы systemctl restart sshd (service sshd restart), верните аутентификацию по паролю, выполните reload службы systemctl reload sshd (services sshd reload). В чём различие между действиями restart и reload? Создайте файл при помощи команды cat > file_name, напишите текст и завершите комбинацией ctrl+d. Какой сигнал передадим процессу?

4. Сигналы процессам. Запустите mc. Используя ps, найдите PID процесса, завершите процесс, передав ему сигнал 9.

5. Степик. Раздел 2. Управление памятью. Программа курса · Операционные системы · Stepik

Решение

1) futitisme@futitismeserver:~$ echo > file.txt
   futitisme@futitismeserver:~$ ls -1
   file.txt
   y
   y.pub
   futitisme@futitismeserver:~$ cat /etc 2> exceptions.txt

2) futitisme@futitismeserver:~$ ls -l /etc | cut -f 1 -d ' '
   total
   -rw-r--r--
   drwxr-xr-x
   ...
   drwxr-xr-x
   -rw-r--r--

   futitisme@futitismeserver:~$ ll -ld /etc | cut -f1 -d " " | sort | uniq
   drwxr-xr-x
   futitisme@futitismeserver:~$ ll -ld /etc/* | cut -f1 -d " " | sort -u
   drwx------

drwxrwxr-x
drwxr-xr-x
lrwxrwxrwx
-r--r-----
-r--r--r--
-rw-r-----
-rw-r--r--
-rwxr-xr-x

3) futitisme@futitismeserver:~$ nano /etc/ssh/sshd_config
futitisme@futitismeserver:~$ systemctl restart sshd
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
==== AUTHENTICATION COMPLETE ===
futitisme@futitismeserver:~$ systemctl reload sshd
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
==== AUTHENTICATION COMPLETE ===
futitisme@futitismeserver:~$ cat > filename
hello world

4)futitisme@futitismeserver:~$ sudo apt-get install mc
futitisme@futitismeserver:~$ mc

futitisme@futitismeserver:~$ ps -ef
futitisme@futitismeserver:~$ killall -s 9 1047

Лабораторная 4. Устройство файловой системы Linux. Понятие Файла и каталога

Задание

1. Создать файл file1 и наполнить его произвольным содержимым. Скопировать его в file2. Создать символическую ссылку file3 на file1. Создать жёсткую ссылку file4 на file1. Посмотреть, какие inode у файлов. Удалить file1. Что стало с остальными созданными файлами? Попробовать вывести их на экран.

2. Дать созданным файлам другие, произвольные имена. Создать новую символическую ссылку. Переместить ссылки в другую директорию.

3. Создать два произвольных файла. Первому присвоить права на чтение и запись для владельца и группы, только на чтение — для всех. Второму присвоить права на чтение и запись только для владельца.

Решение

1) futitisme@futitismeserver:~$ cat > file1
Hello world

futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:50 file1
-rw-rw-r-- 1 futitisme futitisme 12 окт 5 15:19 filename
-rw-rw-r-- 1 futitisme futitisme 1 окт 5 07:35 file.txt
futitisme@futitismeserver:~$ cp file1 file2
futitisme@futitismeserver:~$ cat file2
Hello world

futitisme@futitismeserver:~$ ln -s file1 file3
futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:50 file1
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:51 file2
lrwxrwxrwx 1 futitisme futitisme 5 окт 5 16:51 file3 -> file1
-rw-rw-r-- 1 futitisme futitisme 12 окт 5 15:19 filename
-rw-rw-r-- 1 futitisme futitisme 1 окт 5 07:35 file.txt
futitisme@futitismeserver:~$ ln file1 file4
futitisme@futitismeserver:~$ ll -i file*
147137 -rw-rw-r-- 2 futitisme futitisme 13 окт 5 16:50 file1
147248 -rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:51 file2
147249 lrwxrwxrwx 1 futitisme futitisme 5 окт 5 16:51 file3 -> file1
147137 -rw-rw-r-- 2 futitisme futitisme 13 окт 5 16:50 file4
146810 -rw-rw-r-- 1 futitisme futitisme 12 окт 5 15:19 filename
136325 -rw-rw-r-- 1 futitisme futitisme 1 окт 5 07:35 file.txt
futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 2 futitisme futitisme 13 окт 5 16:50 file1
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:51 file2

lrwxrwxrwx 1 futitisme futitisme 5 окт 5 16:51 file3 -> file1
-rw-rw-r-- 2 futitisme futitisme 13 окт 5 16:50 file4
-rw-rw-r-- 1 futitisme futitisme 12 окт 5 15:19 filename
-rw-rw-r-- 1 futitisme futitisme 1 окт 5 07:35 file.txt
futitisme@futitismeserver:~$ rm file1
futitisme@futitismeserver:~$ ll -i file*
147248 -rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:51 file2
147249 lrwxrwxrwx 1 futitisme futitisme 5 окт 5 16:51 file3 -> file1
147137 -rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:50 file4
146810 -rw-rw-r-- 1 futitisme futitisme 12 окт 5 15:19 filename
136325 -rw-rw-r-- 1 futitisme futitisme 1 окт 5 07:35 file.txt
futitisme@futitismeserver:~$ cat file1
cat: file1: No such file or directory
futitisme@futitismeserver:~$ cat file2
Hello world

futitisme@futitismeserver:~$ cat file3
cat: file3: No such file or directory
futitisme@futitismeserver:~$ cat file4
Hello world

2) futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:51 file2
lrwxrwxrwx 1 futitisme futitisme 5 окт 5 16:51 file3 -> file1
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:50 file4
-rw-rw-r-- 1 futitisme futitisme 12 окт 5 15:19 filename
-rw-rw-r-- 1 futitisme futitisme 1 окт 5 07:35 file.txt
futitisme@futitismeserver:~$ rm file3
futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:51 file2
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:50 file4
-rw-rw-r-- 1 futitisme futitisme 12 окт 5 15:19 filename
-rw-rw-r-- 1 futitisme futitisme 1 окт 5 07:35 file.txt
futitisme@futitismeserver:~$ mv file2 file2_2
futitisme@futitismeserver:~$ mv file4 file4_2
futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:51 file2_2
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:50 file4_2
-rw-rw-r-- 1 futitisme futitisme 12 окт 5 15:19 filename
-rw-rw-r-- 1 futitisme futitisme 1 окт 5 07:35 file.txt
futitisme@futitismeserver:~$ ln -s file2_2 link.rel
futitisme@futitismeserver:~$ ln -s /home/futitisme/file4_2 link.abs
futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:51 file2_2
-rw-rw-r-- 1 futitisme futitisme 13 окт 5 16:50 file4_2
-rw-rw-r-- 1 futitisme futitisme 12 окт 5 15:19 filename
-rw-rw-r-- 1 futitisme futitisme 1 окт 5 07:35 file.txt

```
futitisme@futitismeserver:~$ ln  -s file2_2 filesymlink.rel
futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme 13 окт  5 16:51 file2_2
-rw-rw-r-- 1 futitisme futitisme 13 окт  5 16:50 file4_2
-rw-rw-r-- 1 futitisme futitisme 12 окт  5 15:19 filename
lrwxrwxrwx 1 futitisme futitisme  7 окт  5 17:10 filesymlink.rel -> file2_2
-rw-rw-r-- 1 futitisme futitisme  1 окт  5 07:35 file.txt
futitisme@futitismeserver:~$ ln  -s /home/futitisme/file4_2 filesymlink.abs
futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme 13 окт  5 16:51 file2_2
-rw-rw-r-- 1 futitisme futitisme 13 окт  5 16:50 file4_2
-rw-rw-r-- 1 futitisme futitisme 12 окт  5 15:19 filename
lrwxrwxrwx 1 futitisme futitisme 23 окт  5 17:11 filesymlink.abs -> /home/futitisme/file4_2
lrwxrwxrwx 1 futitisme futitisme  7 окт  5 17:10 filesymlink.rel -> file2_2
-rw-rw-r-- 1 futitisme futitisme  1 окт  5 07:35 file.txt
futitisme@futitismeserver:~$ mkdir dir0
futitisme@futitismeserver:~$ mv filesymlink.* dir0/
futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme 13 окт  5 16:51 file2_2
-rw-rw-r-- 1 futitisme futitisme 13 окт  5 16:50 file4_2
-rw-rw-r-- 1 futitisme futitisme 12 окт  5 15:19 filename
-rw-rw-r-- 1 futitisme futitisme  1 окт  5 07:35 file.txt
futitisme@futitismeserver:~$ ll dir0
total 8
drwxrwxr-x 2 futitisme futitisme 4096 окт  5 17:12 ./
drwxr-x--- 7 futitisme futitisme 4096 окт  5 17:12 ../
lrwxrwxrwx 1 futitisme futitisme   23 окт  5 17:11 filesymlink.abs -> /home/futitisme/file4_2
lrwxrwxrwx 1 futitisme futitisme    7 окт  5 17:10 filesymlink.rel -> file2_2
futitisme@futitismeserver:~$ cat dir0/filesymlink.rel
cat: dir0/filesymlink.rel: No such file or directory
```

3)
```
futitisme@futitismeserver:~$ touch file1_3 file2_3
futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme  0 окт  5 17:15 file1_3
-rw-rw-r-- 1 futitisme futitisme 13 окт  5 16:51 file2_2
-rw-rw-r-- 1 futitisme futitisme  0 окт  5 17:15 file2_3
-rw-rw-r-- 1 futitisme futitisme 13 окт  5 16:50 file4_2
-rw-rw-r-- 1 futitisme futitisme 12 окт  5 15:19 filename
-rw-rw-r-- 1 futitisme futitisme  1 окт  5 07:35 file.txt
futitisme@futitismeserver:~$ chmod a-rwx file*
futitisme@futitismeserver:~$ ll file*
---------- 1 futitisme futitisme  0 окт  5 17:15 file1_3
---------- 1 futitisme futitisme 13 окт  5 16:51 file2_2
---------- 1 futitisme futitisme  0 окт  5 17:15 file2_3
---------- 1 futitisme futitisme 13 окт  5 16:50 file4_2
---------- 1 futitisme futitisme 12 окт  5 15:19 filename
---------- 1 futitisme futitisme  1 окт  5 07:35 file.txt
```

```
futitisme@futitismeserver:~$ chmod 664 file1_3
futitisme@futitismeserver:~$ chmod 600 file2_3
futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme  0 окт  5 17:15 file1_3
---------- 1 futitisme futitisme 13 окт  5 16:51 file2_2
-rw------- 1 futitisme futitisme  0 окт  5 17:15 file2_3
---------- 1 futitisme futitisme 13 окт  5 16:50 file4_2
---------- 1 futitisme futitisme 12 окт  5 15:19 filename
---------- 1 futitisme futitisme  1 окт  5 07:35 file.txt
futitisme@futitismeserver:~$ chmod ug+rw,o+r file1_3
futitisme@futitismeserver:~$ chmod u+rw file2_3
futitisme@futitismeserver:~$ ll file*
-rw-rw-r-- 1 futitisme futitisme  0 окт  5 17:15 file1_3
---------- 1 futitisme futitisme 13 окт  5 16:51 file2_2
-rw------- 1 futitisme futitisme  0 окт  5 17:15 file2_3
---------- 1 futitisme futitisme 13 окт  5 16:50 file4_2
---------- 1 futitisme futitisme 12 окт  5 15:19 filename
---------- 1 futitisme futitisme  1 окт  5 07:35 file.txt
```

Лабораторная 5. Контроль заданий на Степике.

1.2. Раздел Ассемблер 1.

Задание 1.

Напишите код, который обменивает значения в регистрах RSI и RDX. Т. е. если перед выполнением вашего кода в регистре RSI хранится число 1, а в регистре RDX хранится число 2, то после выполнения кода в регистре RSI должно храниться число 2, а в регистре RDX число 1.

Вам разрешено пользоваться следующими регистрами общего назначения: RAX, RBX, RCX, RDX, RBP, RDI, RSI, R8 - R15.

В задании не предполагается использование стека, даже если вы знаете, что это такое.

Решение

```
movq %RDX , %RBX
movq %RSI , %RDX
movq %RBX , %RSI
```

Задание 2.

Сложите два числа в регистрах RSI и RDX, результат должен быть в регистре RSI. Вам разрешено пользоваться следующими регистрами общего назначения: RAX, RBX, RCX, RDX, RBP, RDI, RSI, R8 - R15. В задании не предполагается использование стека, даже если вы знаете, что это такое.

Решение

```
addq %RDX, %RSI
```

Задание 3.

В регистре RSI вам дано целое число - градусы по шкале Фаренгейта (TFT_F) Напишите код, который получит в регистре RSI соответствующее значение по шкале Цельсия (TCT_C).

Формула для перевода из шкалы Фаренгейта в шкалу Цельсия: $TC=5\times(TF-32)/9$

ВАЖНО: в качестве операнда инструкций div и mul не может выступать просто число (a. k. a. immediate), т. е. вам придется загрузить делитель/множитель в регистр и использовать регистр.

Так как мы не рассматривали операции с нецелыми числами, нужно отбросить не целую часть результата и вернуть только целую. Кроме того гарантируется, что входное число и ответ будут беззнаковыми числами (более формально гарантируется, что $32 \le TF \le 220$

Вам разрешено пользоваться следующими регистрами общего назначения: RAX, RBX, RCX, RDX, RBP, RDI, RSI, R8 - R15.

В задании не предполагается использование стека, даже если вы знаете, что это такое.

Решение

```
sub $32, %RSI
movq %RSI, %RAX
movq $5, %RBX
movq $9, %RCX
mulq %RBX
divq %RCX
movq %RAX, %RSI
```

Задание 4.

4. В регистрах RSI и RDX вам даны два числа, ваша задача поменять их местами, как и в одном из предыдущих заданий. Но добавляется условие, что все остальные регистры общего назначения должны остаться неизменными. Т. е. если вы используете какой-то регистр общего назначения кроме RSI и RDX, то вы должны сохранить и затем восстановить сохраненное значение регистра (ну или не пользоваться этими регистрами вовсе).

   Вам разрешено пользоваться следующими регистрами общего назначения: RAX, RBX, RCX, RDX, RBP, RDI, RSI, R8 - R15. В задании предполагается использование стека, но вы должны восстановить стек в исходное состояние.

Решение

```
pushq %RSI
movq %RDX, %RSI
popq %RDX
```

1.3 Язык ассемблера 2

Задание 1.

А теперь ваша задача написать функцию swap. Функция принимает на вход два указателя на 64-битные числа (в регистрах RDI и RSI) и должна обменять значения в памяти.

ВАЖНО: функция принимает указатели на значения, которые нужно обменять, а не сами значения, т. е. не забудьте, что чтобы добраться до значений указатели нужно разыменовать.

Решение

```
swap:
   pushq (%RSI)
   pushq (%RDI)
   popq (%RSI)
   popq (%RDI)
   retq
```

Задание 2.

Попробуйте по аналогии с примером в лекции написать функцию, возвращающую минимум из двух переданных ей аргументов. Аргументы (беззнаковые целые числа) передаются в регистрах RDI и RSI. Результат работы функции должен быть сохранен в регистре RAX.

Решение

```
min:
   // Put your code here rdi, rsi,  rax
   movq %rdi, %rax
   cmpq %rdi, %rsi
   ja get
   movq %rsi, %rax
get:
   ret
```

Задание 3.

Теперь более сложное задание. Вам требуется написать функцию pow, которая принимает на вход два беззнаковых числа x и p (в регистрах RDI и RSI соответственно) и возвращает значение x в степени p в регистре RAX. Гарантируется, что x и p не могут быть равны 0 одновременно (по отдельности они все еще могут быть равны 0). Также гарантируется, что ответ помещается в 64 бита. **ВАЖНО:** не забудьте, что кроме инструкций

условного перехода (jcc) есть и инструкция безусловного перехода jmp - она может вам пригодиться.

Решение

```
pow:
movq $1, %rax
Cycle:
cmpq $0, %rsi
je st1
mulq %rdi
decq %rsi
jmp Cycle
st1:
retq
```

2.4. Управления памятью. Страничная организация памяти.

Проверим, как вы поняли paging. Для этого вам предлагается выступить в качестве процессора и преобразовать несколько логических адресов в физические. Формат входных данных следующий:

- в первой строке вам даны 3 числа $m,q,r \geq 0$, где ***q*** - это количество запросов, на которые вам нужно ответить, ***r*** - физический адрес корневой таблицы страниц
- следующих ***m*** строках записаны пары ***paddr*** и ***value*** - описание физической памяти, каждая пара значит, что по физическому адресу ***paddr*** хранится 64 битное значение ***value***, при этом гарантируется, что все ***paddr*** различны, выровнены на границу 8 байт и помещаются в 64 бита
- в последних ***q*** строках идут целые числа - логические адреса, которые вам нужно преобразовать в физические, для каждого из этих чисел нужно вывести на отдельной строке либо физический адрес, либо слово "***fault***", если преобразовать логический адрес в физический нельзя.

Считайте, что таблица страниц имеет формат 64 битного режима x86 (4 уровня, каждая страница 4 KB, каждая запись 8 байт, формат записи был показан в лекциях), но вы можете игнорировать все поля, кроме бита присутствия (на картинке бит **P** - нулевой бит) и собственно физического адреса.

Для всех физических адресов, не указанных во входных данных (среди ***m*** пар ***paddr value***), считайте, что по этим адресам хранятся нули.

ВАЖНО: это было неочевидно из видео, но все физические адреса, которые хранятся в записях таблицы страниц должны быть **выровнены, как минимум, на границу 4096 байт (4Kb), т. е. младшие 12 бит физических адресов всегда равны 0**, соответственно, хранить младшие биты нет смысла и в записе таблицы страниц они не хранятся - их место занимают специальные флаги. Убедитесь, что вы понимаете приведенный пример.

ВАЖНО2: после каждой неверной попытки вам требуется скачать новый набор данных и использовать его для следующей попытки.

Решение

```
import sys

def get_page(logical_addr: int):
    pml4 = (logical_addr >> 39) & 0x1ff
    dir_ptr = (logical_addr >> 30) & 0x1ff
    directory = (logical_addr >> 21) & 0x1ff
    table = (logical_addr >> 12) & 0x1ff
    offset = logical_addr & 0xfff
    return (pml4, dir_ptr, directory, table, offset)

def get_page_phy_addr(value: int):
    return (value & ((0xfffffffff) << 12))

def get_phy_addr(page: tuple, mem_struct: dict, cr3: int):
    value = cr3
    for i in range(len(page) - 1):
        index = page[i] * 8
        value = mem_struct.get(index + value, 0)
        if value & 1 == 0:
            print("fault")
            return
        value = get_page_phy_addr(value)
    print(value + page[-1])

def main():
    reader = (tuple(map(int, line.split())) for line in sys.stdin)
    mem_rows, queries, cr3 = next(reader)
    mem_struct = dict([next(reader) for _ in range(mem_rows)])
```

```
    for _ in range(queries):
        logical_addr, = next(reader)
        page = get_page(logical_addr)
        get_phy_addr(page, mem_struct, cr3)

if __name__ == "__main__":
    main()
```

2.5. Простой подход к аллокации памяти.

Попробуйте реализовать динамический аллокатор памяти (интерфейс в шаблоне кода). Введем несколько обозначений:

- **BufSize** - размер участка логической памяти, который ваш аллокатор будет распределять
- **MaxSize** - наибольший размер участка памяти, который можно аллоцировать вашим аллокатором для данного **BufSize**(проверяющая система найдет его бинарным поиском)
- **EffectiveSize** - максимальное количество памяти, которое может быть выделено пользователю для данного **BufSize** (например, если мы аллоцируем много небольших участков памяти)

Ваш аллокатор памяти должен удовлетворять следующим условиям:

- **MaxSize** должен быть не меньше **8/9 BufSize**
- **EffectiveSize** должен быть не меньше **1/9 BufSize**
- ваш аллокатор должен бороться с фрагментацией, т. е. если от начального состояния аллокатора мы смогли успешно аллоцировать какое-то количество памяти, то если мы освободим всю эту память и заново попробуем повторить аллокацию, она должна быть успешной
- если аллокатор не смог аллоцировать участок памяти нужного размера, то он должен вернуть NULL
- использование динамической аллокации памяти запрещено (malloc/new/new[]/ free/delete/delete[]).

Гарантируется

- что **BufSize** будет не меньше **100Kb** и не больше **1Mb**

- что минимальный аллоцируемый участок памяти будет не меньше 16 байт.

Решение

```
#include <iostream>
#include <climits>

#ifdef MY_TEST
  #include <cstdlib>
  #include <iomanip>

  #define BUF_SIZE 102400

  #define MY_DEBUG 1
#endif

////////////////////////////////////////////////////////////////////////////
// Basic definitions
// Minimum actual size of a block
#define MIN_BLOCK_SIZE 16
// function to move to the next header (see, getNext() and getPrevious())
typedef struct header* (*HeaderIterator)(struct header*);
// start marker of the block
struct header {
  unsigned char free; // block is free to allocate
  std::size_t actualSize; // size to use. without header and tail
};
// end marker of the block
struct tail {
  unsigned char free; // block is free to allocate
  std::size_t actualSize; // size to use. without header and tail
};

////////////////////////////////////////////////////////////////////////////
// Globals variables
struct header* _startHeader; // the first header
struct tail* _endTail; // the end tail
std::size_t _size; // size of the entire block
std::size_t minSize; // minimum block size
////////////////////////////////////////////////////////////////////////////
// Function
// Mark initial bufer with header and tail
struct header* initBlock(void *buf, std::size_t size) {
  #ifdef MY_DEBUG
    std::cout << "initBlock(" << buf << ", " << size << ')' << std::endl;
  #endif
```

```
    struct header* ph = (struct header*)buf;
    ph->free = 1;
    ph->actualSize = size - sizeof(struct header) - sizeof(struct tail);
    #ifdef MY_DEBUG
      std::cout << "   ph addr =" << ph << "; free =" << (int)(ph->free) << "; actualSize = " << ph-
>actualSize << std::endl;
    #endif

    struct tail* pt = (struct tail*)((unsigned char*)buf + size - sizeof(struct tail));
    pt->free = 1;
    pt->actualSize = ph->actualSize;
    #ifdef MY_DEBUG
      std::cout << "   pt addr =" << pt << "; free =" << (int)(pt->free) << "; actualSize = " << pt-
>actualSize << std::endl;
    #endif

    return ph;
  }
  // get tail corresponds to the header
  struct tail* getTail(struct header* phead) {
    unsigned char* base = (unsigned char*)phead;
    base += sizeof(struct header) + phead->actualSize;
    return (struct tail*)base;
  }
  // get header corresponds to the tail
  struct header* getHeader(struct tail* ptail) {
    unsigned char* base = (unsigned char*)ptail;
    base -= sizeof(struct header) + ptail->actualSize;
    return (struct header*)base;
  }
  // return pointer to the next block.
  // if the current block is the very last, then return NULL
  struct header* getNext(struct header* phead) {
    #ifdef MY_DEBUG
      std::cout << "getNext(" << phead << ')' << std::endl;
    #endif

    struct tail* ptail = getTail(phead);
    if(ptail == _endTail)
      return NULL;

    unsigned char* base = (unsigned char*)ptail;
    base += sizeof(struct tail);
    return (struct header*)base;
  }
  // return pointer to the previous block.
  // if the current block is the very first, then return NULL
```

```
struct header* getPrevious(struct header* phead) {
 #ifdef MY_DEBUG
  std::cout << "getPrevious(" << phead << ')' << std::endl;
 #endif

 if(phead == _startHeader)
  return NULL;

 unsigned char* base = (unsigned char*)phead;
 base -= sizeof(struct tail);
 return getHeader((struct tail*)base);
}
// return size nedeed to allocate block with header and tail
std::size_t getAllSize(std::size_t size) {
 return size + sizeof(struct header) + sizeof(struct tail);
}
// return actual size of the block between header and tail.
std::size_t getActualSize(struct header* ph, struct tail* pt) {
 if((unsigned char*)ph >= (unsigned char*)pt)
  return 0;
 return (unsigned char*)pt - (unsigned char*)ph - sizeof(struct header);
}
// set phStart header and phEnd tail actualSize
//  == (phStart->actualSize + phEnd->actualSize).
// So phStart and ptEnd make new big block.
// Do not check any limits.
void joinBlocks(struct header* phStart, struct header* phEnd) {
 if(phStart == phEnd)
  return;

 #ifdef MY_DEBUG
  std::cout << "joinBlocks(" << phStart << ", " << phEnd << ')' << std::endl;
 #endif

 if(phStart > phEnd) {

  // phStart has to precede to phEnd
  struct header* tmp = phStart;
  phStart = phEnd;
  phEnd = tmp;
  #ifdef MY_DEBUG
   std::cout << "  swap block. phtStart " << phStart << "; phEnd " << phEnd << std::endl;
  #endif
 }

 struct tail* ptEnd = getTail(phEnd);
 #ifdef MY_DEBUG
```

```
  std::cout << "   ptEnd = " << ptEnd << std::endl;
 #endif

 phStart->actualSize = getActualSize(phStart, ptEnd);
 ptEnd->actualSize = phStart->actualSize;
 #ifdef MY_DEBUG
  std::cout << "   actualSize = " << phStart->actualSize << std::endl;
 #endif
}
// Split free block on two parts. First block is free.
// Second is used with size (size).
// Return pointer to the header of second block.
// If there was an error or size of old block too small,
// or the old block is not free return NULL.
struct header* utilizeBlock(struct header* ph, std::size_t size) {
 if(!ph->free || ph->actualSize < size)
  return NULL;

 #ifdef MY_DEBUG
  std::cout << "utilizeBlock(" << ph << ", " << size << ")" << std::endl;
 #endif

 // size with header and tail
 std::size_t allSize = getAllSize(size);

 struct tail* ptEnd = getTail(ph);

 if(ph->actualSize <= allSize) {
  ph->free = 0;
  ptEnd->free = 0;
  return ph;
 }

 std::size_t newSize = ph->actualSize - allSize;

 if(newSize < MIN_BLOCK_SIZE) {
  // it does not make sens to divide the block
  ph->free = 0;
  ptEnd->free = 0;
  return ph;
 }

 // split block on two: Start and End.
 // Because of adjusting, we may take anoter size of the blocks.
 // Compute it.
 unsigned char* base = (unsigned char*)ptEnd;
 base -= allSize;
```

```
  // tail of the start block
  struct tail* ptStart = (struct tail*)base;
  ptStart->free = 1;
  // find new size of the Start block in memory with adjusting
  ptStart->actualSize = getActualSize(ph, ptStart);
  ph->actualSize = ptStart->actualSize;

  // Create header of the second block next the tail of the first block
  struct header* phEnd = (struct header*)(base + sizeof(struct tail));
  phEnd->free = 0;
  phEnd->actualSize = getActualSize(phEnd, ptEnd);

  // The end of the start block became end of the end block
  ptEnd->free = 0;
  ptEnd->actualSize = phEnd->actualSize;

  return phEnd;
}
// Find all consequent free blocks and join it.
// ph - start block header.
// iterator - function to get next block.
// Return pointer to the header of new start block.
struct header* joinNearestFreeBlocks(struct header* ph, HeaderIterator iterator) {
  if(ph == NULL || !ph->free)
    return ph;

  #ifdef MY_DEBUG
    std::cout << "joinNearestFreeBlocks(" << ph << ')' << std::endl;
  #endif

  struct header* next = iterator(ph);
  #ifdef MY_DEBUG
    std::cout << "   next header = " << next << (next == NULL ? '=': '!') << "= NULL";
  #endif

  while(next != NULL && next->free) {
    #ifdef MY_DEBUG
      std::cout << "; free = " << (unsigned)(next->free) << std::endl;
    #endif
    joinBlocks(ph, next); // it does not need to ph precede next
    ph = next;
    next = iterator(ph);
    #ifdef MY_DEBUG
      std::cout << "   next header = " << next << (next == NULL ? '=': '!') << "= NULL";
    #endif
  }
```

```
 #ifdef MY_DEBUG
  std::cout << std::endl;
 #endif

 return ph;
}
  // Эта функция будет вызвана перед тем как вызывать myalloc и myfree
  // используйте ее чтобы инициализировать ваш аллокатор перед началом
  // работы.
  //
  // buf - указатель на участок логической памяти, который ваш аллокатор
  //       должен распределять, все возвращаемые указатели должны быть
  //       либо равны NULL, либо быть из этого участка памяти
  // size - размер участка памяти, на который указывает buf
  void mysetup(void *buf, std::size_t size) {
  _size = size;
  _startHeader = initBlock(buf, size);
  _endTail = getTail(_startHeader);
  // minimum size of the block
  minSize = getAllSize(MIN_BLOCK_SIZE);
 }
  // Функция аллокации
  void *myalloc(std::size_t size) {
  // look up for free block with apropriate size
  std::size_t allSize = getAllSize(size);
  struct header* ph = _startHeader;
  while(ph != NULL) {
   if(ph->free && (ph->actualSize >= allSize || ph->actualSize >= size)) {
    // we find free block with needed size
    unsigned char *base = (unsigned char*)utilizeBlock(ph, size);
    base += sizeof(struct header);
    return (void*)base;
   }
   ph = getNext(ph);
  }
  // there is not any free block with needed size
  return NULL;
 }
  // Функция освобождения
  void myfree(void *p) {
  #ifdef MY_DEBUG
   std::cout << "myfree(" << p << ')' << std::endl;
  #endif

  if(p == NULL)
   return;
```

```
        struct header* ph = (struct header*)((unsigned char*)p - sizeof(struct header));
        ph->free = 1;

        #ifdef MY_DEBUG
          std::cout << "   ph addr =" << ph << "; free =" << (unsigned int)(ph->free) << "; actualSize = "
<< ph->actualSize << std::endl;
        #endif

        struct tail* pt = getTail(ph);
        pt->free = 1;

        #ifdef MY_DEBUG
          std::cout << "   pt addr =" << pt << "; free =" << (unsigned int)(pt->free) << "; actualSize = "
<< pt->actualSize << std::endl;
        #endif

        // join blocks prevous to the current
        ph = joinNearestFreeBlocks(ph, getPrevious);
        // join blocks next to the current
        joinNearestFreeBlocks(ph, getNext);
      }
```

2.7 SLAB аллокатор

Попробуйте реализовать кеширующий аллокатор. Детали интерфейса, который вам нужно реализовать, вы можете найти в шаблоне кода, как и объявления функций, которые вы должны использовать для аллокации SLAB-ов. Кроме предоставленных функций аллокации никакими **другими способами динамической аллокации памяти пользоваться нельзя**.

Считайте, что для аллокации SLAB-ов используется buddy аллокатор (с соответствующей алгоритмической сложностью и ограничениями). Гарантируется, что **возвращаемый указатель будет выровнен на размер аллоцируемого участка** (т. е. если вы аллоцируете SLAB размером 4Kb, то его адрес будет выровнен на границу 4Kb, если 8Kb, то на границу 8Kb и тд).

При реализации вам не обязательно точно следовать рассказанному в видео или описанному в статье подходах. Но вы должны учитывать, что, среди прочего, **проверяющая система будет оценивать работу функции cache_shrink**. При оценке проверяющая система будет считать, что если все аллоцированные из некоторого SLAB-а объекты были освобождены к моменту вызова cache_shrink, то **cache_shrink должен освободить этот SLAB**. Т. е. другими словами, cache_shrink должен возвращать все свободные SLAB-ы системе.

Решение

```
#include <inttypes.h>
```

```
#include <stdlib.h>
/**
 * Эти две функции вы должны использовать для аллокации
 * и освобождения памяти в этом задании. Считайте, что
 * внутри они используют buddy аллокатор с размером
 * страницы равным 4096 байтам.
 **/
/**
 * Аллоцирует участок размером 4096 * 2^order байт,
 * выровненный на границу 4096 * 2^order байт. order
 * должен быть в интервале [0; 10] (обе границы
 * включительно), т. е. вы не можете аллоцировать больше
 * 4Mb за раз.
 **/
void *alloc_slab(int order);
/**
 * Освобождает участок ранее аллоцированный с помощью
 * функции alloc_slab.
 **/
void free_slab(void *slab);
typedef struct memory_block
{
   struct memory_block *next;
} block_mem_t;
typedef struct slab_header
{
   block_mem_t *blocks;
   size_t free_num;
   struct slab_header *next;
   struct slab_header *prev;
} slab_head_t;
#define SLAB_OBJECTS_MIN_NUM 100
/**
 * Эта структура представляет аллокатор, вы можете менять
 * ее как вам удобно. Приведенные в ней поля и комментарии
 * просто дают общую идею того, что вам может понадобится
 * сохранить в этой структуре.
 **/
struct cache {
   slab_head_t *full;
   slab_head_t *partly_full;
   slab_head_t *free;
   size_t object_size; /* размер аллоцируемого объекта */
   int slab_order; /* используемый размер SLAB-а */
   size_t slab_objects; /* количество объектов в одном SLAB-е */
};
static void chache_init_slab(slab_head_t *slab, size_t object_size, size_t objects_num)
```

```
        {
            void *mem = (uint8_t*)slab + sizeof(slab_head_t);
            size_t offset = sizeof(block_mem_t) + object_size;
            for (int i = 0; i < objects_num; ++i)
            {
                ((block_mem_t*)mem)->next = (i + 1) == objects_num ? NULL : (block_mem_t*)
((uint8_t*)mem + offset);
                mem = (uint8_t*)mem + offset;
            }
            slab->blocks = (block_mem_t*)((uint8_t*)slab + sizeof(slab_head_t));
            slab->free_num = objects_num;
        }
        static slab_head_t *cache_create_slab(int order)
        {
            slab_head_t *slab;
            slab = (slab_head_t*)alloc_slab(order);

            slab->blocks = NULL;
            slab->free_num = 0;
            slab->next = NULL;
            return slab;
        }
        /**
         * Функция инициализации будет вызвана перед тем, как
         * использовать это кеширующий аллокатор для аллокации.
         * Параметры:
         *  - cache - структура, которую вы должны инициализировать
         *  - object_size - размер объектов, которые должен
         *    аллоцировать этот кеширующий аллокатор
         **/
        void cache_setup(struct cache *cache, size_t object_size)
        {
            int order = 0;
            size_t meta_size = sizeof(slab_head_t);
            cache->full = NULL;
            cache->partly_full = NULL;
            cache->free = NULL;
            cache->object_size = object_size;
            size_t min_mem_required = sizeof(slab_head_t) + (sizeof(block_mem_t) + object_size) *
SLAB_OBJECTS_MIN_NUM;
            while (((1UL << order) * 4096) < min_mem_required)
            {
                ++order;
            }
            cache->slab_order = order;
            size_t addition_mem = ((1UL << order) * 4096 / min_mem_required - 1) * min_mem_required
+
```

```
                        ((1UL << order) * 4096 % min_mem_required);
    cache->slab_objects = SLAB_OBJECTS_MIN_NUM + addition_mem / (sizeof(block_mem_t)
+ object_size);
}
/**
 * Функция освобождения будет вызвана когда работа с
 * аллокатором будет закончена. Она должна освободить
 * всю память занятую аллокатором. Проверяющая система
 * будет считать ошибкой, если не вся память будет
 * освбождена.
 **/
void cache_release(struct cache *cache)
{
    slab_head_t *tmp;
    tmp = cache->full;
    while (tmp)
    {
        cache->full = tmp->next;
        free_slab(tmp);
        tmp = cache->full;
    }
    tmp = cache->partly_full;
    while (tmp)
    {
        cache->partly_full = tmp->next;
        free_slab(tmp);
        tmp = cache->partly_full;
    }
    tmp = cache->free;
    while (tmp)
    {
        cache->free = tmp->next;
        free_slab(tmp);
        tmp = cache->free;
    }
}
static void cache_move_slab(struct cache *cache, slab_head_t **dest, slab_head_t *src)
{
    if (cache->full == src)
        cache->full = src->next;
    else if (cache->partly_full == src)
        cache->partly_full = src->next;
    else if (cache->free == src)
        cache->free = src->next;
    if (src->prev)
    {
        src->prev->next = src->next;
```

```
    }
    if (src->next)
    {
        src->next->prev = src->prev;
    }
    src->prev = NULL;
    src->next = *dest;
    if (*dest)
        (*dest)->prev = src;
    *dest = src;
}
static void cache_free_block(slab_head_t *slab, block_mem_t *block)
{
    block->next = slab->blocks;
    slab->blocks = block;
    ++slab->free_num;
}
static block_mem_t *cache_alloc_block(slab_head_t *slab)
{
    block_mem_t *block;
    static size_t i = 0;
    block = slab->blocks;
    slab->blocks = block->next;
    --slab->free_num;
    block->next = NULL; // do we need it actually?
    return block;
}
/**
 * Функция аллокации памяти из кеширующего аллокатора.
 * Должна возвращать указатель на участок памяти размера
 * как минимум object_size байт (см cache_setup).
 * Гарантируется, что cache указывает на корректный
 * инициализированный аллокатор.
 **/
void *cache_alloc(struct cache *cache)
{
    block_mem_t *block;
    if (cache->partly_full)
    {
        block = cache_alloc_block(cache->partly_full);

        if (cache->partly_full->free_num == 0)
        {
            cache_move_slab(cache, &cache->full, cache->partly_full);
        }
    }
    else if (cache->free)
```

```
    {
        block = cache_alloc_block(cache->free);
        cache_move_slab(cache, &cache->partly_full, cache->free);
    }
    else
    {
        slab_head_t *slab = cache_create_slab(cache->slab_order);

        chache_init_slab(slab, cache->object_size, cache->slab_objects);
        block = cache_alloc_block(slab);
        cache->partly_full = slab;
    }
    return (uint8_t*)block + sizeof(block_mem_t);
}

/**
 * Функция освобождения памяти назад в кеширующий аллокатор.
 * Гарантируется, что ptr - указатель ранее возвращенный из
 * cache_alloc.
 **/
void cache_free(struct cache *cache, void *ptr)
{
    slab_head_t *slab = (slab_head_t*)((uint64_t)ptr & ~((1UL << cache->slab_order) * 4096 - 1));
    block_mem_t *block = (block_mem_t*)((uint8_t*)ptr - sizeof(block_mem_t));
    cache_free_block(slab, block);
    if (slab->free_num == 1)
    {
        cache_move_slab(cache, &cache->partly_full, slab);
    }
    else if (slab->free_num == cache->slab_objects)
    {
        cache_move_slab(cache, &cache->free, slab);
    }
}

/**
 * Функция должна освободить все SLAB, которые не содержат
 * занятых объектов. Если SLAB не использовался для аллокации
 * объектов (например, если вы выделяли с помощью alloc_slab
 * память для внутренних нужд вашего алгоритма), то освбождать
 * его не обязательно.
 **/
void cache_shrink(struct cache *cache)
{
    slab_head_t *tmp;
    tmp = cache->free;
    while (tmp)
```

```
    {
        cache->free = tmp->next;
        free_slab(tmp);
        tmp = cache->free;
    }
}
```

3.4 Планирование и многозадачность. Критерии планирования

Попробуйте решить задачу в общем случае. В первой строке вам дается число задач N.

В следующей строке идет описание задач, для каждой задачи вам дана ее продолжительность - T_i (где i - это номер задачи, от 0 до $N-1$ не включительно). На выход вам требуется вывести номера задач (задачи нумеруются с 0) в порядке, который минимизирует среднее время ожидания завершения задачи, как это было объяснено ранее.

Гарантируется, что $0<N\leq100000$ и $0<Ti\leq100000$ и все числа целые.

Решение

```
n = input()

print(" ".join(map(lambda t: str(t[0]),
sorted(enumerate(map(int, input().split())),
key=lambda t: t[1]))))
```

3.5. Планирование и многозадачность. Реалистичное планирование

В этом задании вам потребуется реализовать планировщик, использующий алгоритм Round Robin. Реализация планировщика состоит из нескольких функций:

- **void scheduler_setup(int timeslice)** - вызывается перед началом работы, а timeslice - квант времени, который нужно использовать в некоторых единицах времени (что именно используется как единица времени, не существенно);
- **void new_thread(int thread_id)** - оповещает планировщик о новом потоке с идентификатором thread_id;

- **void exit_thread()** - оповещает планировщик о том, что текущий исполняемый на CPU поток завершился (соответственно, планировщик должен отдать CPU кому-то другому);
- **void block_thread()** - оповещает планировщик, что текущий исполняемый поток был заблокирован (например, запросил IO операцию и должен отдать CPU);
- **void wake_thread(int thread_id)** - оповещает, что поток с идентификатором thread_id был разблокирован (например, IO операция завершилась);
- **void timer_tick()** - вызывается через равные интервалы времени, нотифицирует, что прошла одна единица времени;
- **int current_thread(void)** - функция должна возвращать идентификатор потока, который сейчас должен выполняться на CPU, если такого потока нет, то нужно вернуть -1.

При выполнении задания каждый раз, когда поток выполняется на CPU и вызывается **timer_tick**, считайте, что поток отработал целую единицу времени на CPU. Т. е. даже если предыдущий поток добровольно освободил CPU (вызвав **block_thread** или **exit_thread**) и сразу после того, как CPU был отдан другому потоку, была вызвана функция **timer_tick**, то все равно считается, что второй поток отработал целую единицу времени на CPU.

Решение

```
#include <queue>
static std::queue<int> *_threads_queue;
static int _timeslice;
static int _ticked;
static int _current_thread_id;
void switch_to_next_thread()
{
   _ticked = 0;
   if (_threads_queue->size())
   {
      _current_thread_id = _threads_queue->front();
      _threads_queue->pop();
   }
   else
```

```
    {
        _current_thread_id = -1;
    }
}
void scheduler_setup(int timeslice)
{
    _threads_queue = new std::queue<int>();
    _timeslice = timeslice;
    _ticked = 0;
    _current_thread_id = -1;
}

void new_thread(int thread_id)
{
    if (_current_thread_id == -1)
    {
        _current_thread_id = thread_id;
    }
    else
    {
        _threads_queue->push(thread_id);
    }
}

void exit_thread()
{
    switch_to_next_thread();
}

void block_thread()
{
    switch_to_next_thread();
}

void wake_thread(int thread_id)
{
    if (_current_thread_id == -1)
    {
        _current_thread_id = thread_id;
    }
    else
    {
        _threads_queue->push(thread_id);
    }
}

void timer_tick()
```

```
{
   if (_current_thread_id == -1)
   {
      return;
   }
   if (++_ticked == _timeslice)
   {
      _threads_queue->push(_current_thread_id);
      switch_to_next_thread();
   }
}

int current_thread()
{
   return _current_thread_id;
}
```

4.4 Взаимное исключение с использованием RMW регистров

Вам даны две функции: load_linked и store_conditional (объявления и еще одно объяснение принципа работы даны в комментариях к коду). С помощью этих функций реализуйте атомарный инкремент и CAS операцию (объявления и описание логики работы даны в комментариях к коду).

Решение

```
int atomic_fetch_add(atomic_int *x, int arg)
{
   /* Ваш код здесь */
   for (;;) {
      int old_value = load_linked(x);
      if (store_conditional(x, old_value + arg))
         return old_value;
   }
}
bool atomic_compare_exchange(atomic_int *x, int *expected_value,
                        int new_value)
{
   /* Ваш код здесь */
   for (;;) {
      int old_value = load_linked(x);
      if (old_value != *expected_value) {
         *expected_value = old_value;
         return false;
      }
      if (store_conditional(x, new_value))
         return true;
   }
```

```
}
```

4.6 Deadlock-и и средства борьбы с ними

Тяжелое задание этой недели. Реализовать Wait/Die блокировки, которые позволяют избежать deadlock-ов.

Решать это задание можно только на языке C. Вам будут доступны блокировки (struct lock), переменные состояния (struct condition) и атомарные Read/Modify/Write регистры (atomic_int, atomic_uint, atomic_short, atomic_ushort, atomic_long, atomic_ulong, atomic_llong, atomic_ullong) с интерфейсом, который использовался в видео.

Проверяющая система будет оценивать вашу реализацию на ряде простых однопоточных тестов:

- если только один поток пытается захватить блокировку, то захват должен быть успешным
- попытка захвата одной блокировки несколько раз из одного потока не должна приводить к deadlock-у

Кроме простых однопоточных тестов, естественно, будет и многопоточный: несколько потоков будут пытаться захватить блокировки в таком порядке, который с большой вероятностью приведет к deadlock-у. Если ваше решение не укладывается в лимит времени, значит в вашем коде скорее всего случился deadlock.

Требуемый интерфейс реализации описан в шаблоне кода.

Решение

```
struct lock;
void lock_init(struct lock *lock);
void lock(struct lock *lock);
void unlock(struct lock *lock);
struct condition;
void condition_init(struct condition *cv);
void wait(struct condition *cv, struct lock *lock);
void notify_one(struct condition *cv);
void notify_all(struct condition *cv);
struct wdlock_ctx;
struct wdlock {
   struct wdlock *next;
```

```
    const struct wdlock_ctx *owner;

    struct lock lock;
    struct condition cv;
};

struct wdlock_ctx {
    unsigned long long timestamp;
    struct wdlock *locks;
};

void wdlock_ctx_init(struct wdlock_ctx *ctx)
{
    static atomic_ullong next;
    ctx->timestamp = atomic_fetch_add(&next, 1) + 1;
    ctx->locks = NULL;
}
void wdlock_init(struct wdlock *lock)
{
    lock_init(&lock->lock);
    condition_init(&lock->cv);
    lock->owner = NULL;
}
int wdlock_lock(struct wdlock *l, struct wdlock_ctx *ctx)
{
    lock(&l->lock);
    while (l->owner != NULL) {
        if (l->owner->timestamp <= ctx->timestamp) {
            unlock(&l->lock);
            return 0;
        }
        wait(&l->cv, &l->lock);
    }
    l->owner = ctx;
    l->next = ctx->locks;
    ctx->locks = l;
    unlock(&l->lock);
    return 1;
}
void wdlock_unlock(struct wdlock_ctx *ctx)
{
    while (ctx->locks != NULL) {
        struct wdlock *tmp = ctx->locks;
        lock(&tmp->lock);
        tmp->owner = NULL;
        ctx->locks = tmp->next;
        tmp->next = NULL;
```

```
            notify_all(&tmp->cv);
            unlock(&tmp->lock);
        }
    }
```

5.1. Исполняемые файлы и процессы

Задание 1.

Напишите функцию, которая по заданному имени ELF файла возвращает адрес точки входа.

Решение

```
#include <gelf.h>
#include <fcntl.h>
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

uintptr_t entry_point(const char *filename) {
    int fd = open(filename, O_RDONLY);
    if (fd == -1) {
        perror("open()");
        return 0;
    }

    if (!elf_version(EV_CURRENT)) {
        fprintf(stderr, "elf_version(): %s\n", elf_errmsg(elf_errno()));
        return 0;
    }

    Elf *elf = elf_begin(fd, ELF_C_READ, NULL);
    if (!elf) {
        fprintf(stderr, "elf_begin(): %s\n", elf_errmsg(elf_errno()));
        close(fd);
        return 0;
    }

    GElf_Ehdr ehdr;
    if (!gelf_getehdr(elf, &ehdr)) {
        fprintf(stderr, "gelf_getehdr(): %s\n", elf_errmsg(elf_errno()));
        elf_end(elf);
        close(fd);
        return 0;
```

```
    }

    elf_end(elf);
    close(fd);

    return ehdr.e_entry;
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        return EXIT_FAILURE;
    }

    uintptr_t entry = entry_point(argv[1]);
    if (!entry) {
        return EXIT_FAILURE;
    }

    printf("Entry: %" PRIxPTR "\n", entry);

    return EXIT_SUCCESS;
}
```

Задание 2.

В это задании вам нужно для ELF файла найти все его program header-ы, которые описывают участки памяти (p_type == PT_LOAD), и посчитать, сколько места в памяти нужно, чтобы загрузить программу.

Решение

```
#include <iostream>

#define ELF_NIDENT    16
#define STDIN         1
// program header-ы такого типа нужно загрузить в
// память при загрузке приложения
#define PT_LOAD       1
// структура заголовка ELF файла
struct elf_hdr {
    std::uint8_t e_ident[ELF_NIDENT];
    std::uint16_t e_type;
    std::uint16_t e_machine;
    std::uint32_t e_version;
    std::uint64_t e_entry;
```

```
    std::uint64_t e_phoff;
    std::uint64_t e_shoff;
    std::uint32_t e_flags;
    std::uint16_t e_ehsize;
    std::uint16_t e_phentsize;
    std::uint16_t e_phnum;
    std::uint16_t e_shentsize;
    std::uint16_t e_shnum;
    std::uint16_t e_shstrndx;
} __attribute__((packed));
// структура записи в таблице program header-ов
struct elf_phdr {
    std::uint32_t p_type;
    std::uint32_t p_flags;
    std::uint64_t p_offset;
    std::uint64_t p_vaddr;
    std::uint64_t p_paddr;
    std::uint64_t p_filesz;
    std::uint64_t p_memsz;
    std::uint64_t p_align;
} __attribute__((packed));
std::size_t space(const char *name) {
    FILE * f = fopen(name, "r");
    struct elf_hdr * header = (struct elf_hdr *) malloc(sizeof(struct elf_hdr));
    fread(header, sizeof(struct elf_hdr), STDIN, f);
    struct elf_phdr * sz = (struct elf_phdr *) malloc(header->e_phnum * header->e_phentsize);
    fseek(f, header->e_phoff, SEEK_SET);
    for (size_t i = 0; i < header->e_phnum;i++) {
    fseek(f, header->e_phoff + header->e_phentsize * i, SEEK_SET);
        fread(&sz[i],header->e_phentsize, STDIN, f);
    }
    std::size_t val = 0;
    for (size_t i = 0; i < header->e_phnum; i++) {
        if (sz[i].p_type == PT_LOAD) val += (sz[i].p_memsz);
    }
    return val;
}
```

Лабораторная 5. Дополнительное задание .

Задание

Задания со * не обязательные, и носят факультативный характер. Будут учитываться при начислении дополнительных баллов.

1. * Создать группу developer и нескольких пользователей, входящих в неё. Создать директорию для совместной работы. Сделать так, чтобы созданные одними пользователями файлы могли изменять другие пользователи этой группы.
2. * Создать в директории для совместной работы поддиректорию для обмена файлами, но чтобы удалять файлы могли только их создатели.
3. * Создать директорию, в которой есть несколько файлов. Сделать так, чтобы открыть файлы можно было посмотреть, только зная имя файла, а через ls список файлов посмотреть было нельзя.

Решение

```
futitisme@futitismeserver:~$ sudo groupadd developer
futitisme@futitismeserver:~$ cat /etc/group
developer:x:1013:
futitisme@futitismeserver:~$ sudo adduser developer1
futitisme@futitismeserver:~$ sudo adduser developer2
futitisme@futitismeserver:~$ sudo usermod -aG developer developer1
futitisme@futitismeserver:~$ sudo usermod -aG developer developer2
futitisme@futitismeserver:~$ id developer1
uid=1009(developer1) gid=1014(developer1) groups=1014(developer1),1013(developer)
futitisme@futitismeserver:~$ sudo mkdir -p /developer
futitisme@futitismeserver:~$ sudo chgrp -R developer /developer
futitisme@futitismeserver:~$ sudo chmod -R 2775 /developer
```

Лабораторная 6. Введение в скрипты Bash. Планировщики задач crontab и at

Задание

1. Вывести на экран 3 раза имя пользователя, от которого запускается команда.
2. Вывести с помощью цикла while все четные числа от 0 до 100 включительно.
3. Создать с помощью nano файл test.txt. Настроить автоматический бэкап этого файла раз в 10 минут в файл с названием test.txt.bak с использованием cron.

Решение

futitisme@futitismeserver:~$ for i in {1..3}; do echo $(whoami); done futitisme
futitisme
futitisme
futitisme@futitismeserver:~$ nano while2
futitisme@futitismeserver:~$ cat while2
#!/bin/bash
a=0
while [$a -le 100]
do
echo $a
a=$[$a+2]
done
futitisme@futitismeserver:~$ bash while2 0
2
4
6
8
10
...
100
cat: test.txt: No such file or directory
futitisme@futitismeserver:~$ nano test.txt futitisme@futitismeserver:~$ cat test.txt kaif
lolololololol
futitisme@futitismeserver:~$ crontab -e
no crontab for futitisme - using an empty one Select an editor. To change later, run 'select-editor'.
1. /bin/nano <---- easiest 2. /usr/bin/vim.basic
3. /usr/bin/mcedit
4. /usr/bin/vim.tiny
5. /bin/ed
Choose 1-5 [1]: 1
crontab: installing new crontab futitisme@futitismeserver:~$ touch test.txt.back
futitisme@futitismeserver:~$ cat test.txt.back kaif
lolololololol

Лабораторная 7. Процессы в OS Linux.

Задание

Проанализируйте любые два процесса с использованием утилиты “ps” и “pstree”: определите их дочерние и родительские процессы. Приведите дерево исследуемых процессов.

Контроль выполнения заданий по Степику.

Прислать принтскрин автопроверки к заданиям по двум разделам:

1. Управление памятью. 2.1 - 2.7.
2. Планирование и многозадачность. 3.1-3.5

Решение

futitisme@futitismeserver:~$ **ps**
PID TTY 1272 pts/0 17013 pts/0
TIME CMD 00:00:00 bash
00:00:00 ps futitisme@futitismeserver:~$ **ps -e**
PID TTY TIME CMD
1 ?
2 ?
3 ?
4 ?
5 ?
6 ?
7 ? —————— 16969 ? 16970 ? 17034 pts/0
00:00:00 systemd
00:00:00 kthreadd
00:00:00 rcu_gp
00:00:00 rcu_par_gp
00:00:00 netns
00:00:00 kworker/0:0-events
00:00:00 kworker/0:0H-events_highpri
00:00:00 kworker/u8:2-events_unbound 00:00:00 kworker/0:2-events
PID TTY TIME CMD
00:00:00 ps futitisme@futitismeserver:~$ **ps -d**
2 ? 3 ? 4 ? 5 ?
00:00:00 kthreadd 00:00:00 rcu_gp 00:00:00 rcu_par_gp 00:00:00 netns
futitisme@futitismeserver:~$ **ps 463**
PID TTY STAT TIME COMMAND 463 ? I< 0:00 [kaluad]
futitisme@futitismeserver:~$ **ps -eF**
UID PID PPID C SZ RSS PSR STIME TTY

root 1
root 2
root 3
root 4
root 5
———————
root 6
root 16969
root 16970
futitis+ 17081 1272 0 2357 1648 2 07:38 pts/0 00:00:00 ps -eF futitisme@futitismeserver:~$ **ps -a**
11348 2 07:09 ?
0 0 0 0
0
TIME CMD 00:00:00 bash
17114 pts/0 futitisme@futitismeserver:~$ **ps -u**
PID TTY 1159 tty1
0 0 0 0 2 0 2 0 2 0
41987 0
0
0
0
3 07:09 ? 0 07:09 ? 0 07:09 ? 0 07:09 ?
00:00:00 [kthreadd] 00:00:00 [rcu_gp] 00:00:00 [rcu_par_gp] 00:00:00 [netns]
2 0
2 0
2 0
0
0 07:09 ?
0 3 07:33 ? 0 0 07:33 ?
00:00:00 [kworker/0:0 00:00:00 [kworker/u8: 00:00:00 [kworker/0:2
00:00:00 ps
0 0
TIME CMD 00:00:00 /sbin/init
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND futitis+ 1159 0.0
0.1 8252 5020 tty1 S+ 07:09 0:00 -bash
futitis+ 1272 0.0 0.1 8244 5060 pts/0 Ss 07:09 0:00 -bash
futitis+ 17116 0.0 0.0 9428 1640 pts/0 R+ 07:39 0:00 ps -u
futitisme@futitismeserver:~$ **pstree futitisme**
bash
sshd———bash———pstree
systemd———(sd-pam)
futitisme@futitismeserver:~$ **pstree -g** systemd(1)─┬─/usr/bin/monito(812)———monitorix-httpd(870)
├─ModemManager(758)─┬─{ModemManager}(758) │ └─{ModemManager}(758)
├─agetty(724)
├─cron(694)
├─dbus-daemon(696) ├─irqbalance(705)———{irqbalance}(705) ├─login(727)———bash(1159)
├─multipathd(469)─┬─{multipathd}(469)

- │ ├─{multipathd}(469)
- │ ├─{multipathd}(469)
- │ ├─{multipathd}(469)
- │ ├─{multipathd}(469)
──────── ├─udisksd(716)─┬─{udisksd}(716)
- │ ├─{udisksd}(716)
- │ ├─{udisksd}(716) └─unattended-upgr(766)───{unattended-upgr}(766)

futitisme@futitismeserver:~$ **pstree -s 870** systemd───/usr/bin/monito───monitorix-httpd
futitisme@futitismeserver:~$ **pstree 870** monitorix-httpd

Лабораторная 8. Межпроцессное взаимодействие (IPC).

Задание

1) Создать именованный канал и проверить гипотезу о том, что у каналов может быть только один “читатель”.
2) Показать как работает межпроцессное взаимодействие на уровне сигналов:
   a) вывести таблицу сигналов;
   b) послать сигнал прекращающий работу именованного канала (-ов) в пункте 1).
3) Создать неименнованный канал с использованием | и кратко описать вид такого межпроцессного взаимодействия.

Решение

futitisme@futitismeserver:~$ mkfifo pipe123 futitisme@futitismeserver:~$ ll
prw-rw-r-- 1 futitisme futitisme 0 дек 8 11:01 pipe123| futitisme@futitismeserver:~$ echo "hello" > pipe123 futitisme@futitismeserver:~$ echo "world" > pipe123 futitisme@futitismeserver:~$ tail -f ./pipe123 hello
world ^C
futitisme@futitismeserver:~$ kill -l
1) SIGHUP 2) SIGINT 3) SIGQUIT
6) SIGABRT 7) SIGBUS 8) SIGFPE
11) SIGSEGV 12) SIGUSR2 13) SIGPIPE
16) SIGSTKFLT 17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP
21) SIGTTIN 22) SIGTTOU 23) SIGURG 24) SIGXCPU 25) SIGXFSZ
26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO 30) SIGPWR
31) SIGSYS 34) SIGRTMIN 35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3
38) SIGRTMIN+4 SIGRTMIN+8
43) SIGRTMIN+9 SIGRTMIN+13
48) SIGRTMIN+14 SIGRTMAX-12
53) SIGRTMAX-11 SIGRTMAX-7
58) SIGRTMAX-6 SIGRTMAX-2
63) SIGRTMAX-1
39) SIGRTMIN+5 44) SIGRTMIN+10 49) SIGRTMIN+15 54) SIGRTMAX-10 59) SIGRTMAX-5 64) SIGRTMAX

40) SIGRTMIN+6 45) SIGRTMIN+11 50) SIGRTMAX-14 55) SIGRTMAX-9 60) SIGRTMAX-4
41) SIGRTMIN+7 42) 46) SIGRTMIN+12 47) 51) SIGRTMAX-13 52) 56) SIGRTMAX-8 57) 61) SIGRTMAX-3 62)
futitisme@futitismeserver:~$ ps aux | grep tail
futitis+ 1555 0.0 0.0 5216 820 pts/2 S+ 11:02 0:00 tail -f ./pipe123 futitis+ 1658 0.0 0.0 5892 1980 pts/1 S+ 11:06 0:00 grep --color=auto tail futitisme@futitismeserver:~$ kill -15 1555

futitisme@futitismeserver:~$ tail -f ./pipe123
Terminated
futitisme@futitismeserver:~$ find /usr/ | grep /cron | sort
/usr/bin/crontab
/usr/lib/python3/dist-packages/sos/report/plugins/cron.py /usr/lib/python3/dist-packages/sos/report/plugins/__pycache__/cron.cpython-310.pyc /usr/lib/systemd/system/cron.service
/usr/sbin/cron /usr/share/bash-completion/completions/crontab /usr/share/bug/cron
/usr/share/bug/cron/control /usr/share/bug/cron/script

Лабораторная 9. Управление пакетами и репозиториями. Основы сетевой безопасности

Задание

1. Подключить репозиторий с nginx любым удобным способом, установить nginx и потом удалить nginx, используя утилиту dpkg.

2. Установить пакет на свой выбор, используя snap.

3. Настроить iptables: разрешить подключения только на 22-й и 80-й порты.

4. * Настроить проброс портов локально с порта 80 на порт 8080.

Решение

1)futitisme@futitismeserver:~$ sudo wget https://nginx.org/keys/nginx_signing.key --2022-12-10 08:27:35-- https://nginx.org/keys/nginx_signing.key
Resolving nginx.org (nginx.org)... 3.125.197.172, 52.58.199.22, 2a05:d014:edb:5704::6, ... Connecting to nginx.org (nginx.org)|3.125.197.172|:443... connected.

HTTP request sent, awaiting response... 200 OK Length: 1561 (1,5K) [application/octet-stream] Saving to: ‘nginx_signing.key.2’

nginx_signing.key.2 100%[==================>] 1,52K --.-KB/s in 0s

2022-12-10 08:27:35 (388 MB/s) - ‘nginx_signing.key.2’ saved [1561/1561]
futitisme@futitismeserver:~$ sudo apt-key add nginx_signing.key futitisme@futitismeserver:~$ sudo nano /etc/apt/sources.list futitisme@futitismeserver:~$ sudo systemctl start nginx
futitisme@futitismeserver:~$ sudo dpkg --remove nginx futitisme@futitismeserver:~$ sudo dpkg --purge nginx

2)futitisme@futitismeserver:~$ s**udo snap install** tetris-ever tetris-ever 20.21.11 from Jintao Yang (joker2770) installed futitisme@futitismeserver:~$ **sudo snap remove** tetris-ever tetris-ever removed

3) iptables -P INPUT DROP

sudo iptables -A INPUT -s 80 -j ACCEPT
sudo iptables -A INPUT -s 22 -j ACCEPT

4)futitisme@futitismeserver:~$ sudo iptables -t nat -A PREROUTING -p tcp -- dport 80 -j REDIRECT --to-port 8080

Лабораторная 10. Введение в Docker.

Задание

1. Переустановить операционную систему (по желанию, для дополнительной практики)
2. Подключить репозиторий Docker.
3. Запустить контейнер с Ubuntu.
4. * Используя Dockerfile, собрать связку nginx + PHP-FPM в одном контейнере.

Решение

2. Подключить репозиторий Docker.
utitisme@futitismeserver:~$ **sudo snap install docker**
docker 20.10.17 from Canonical✓ installed
futitisme@futitismeserver:~$ **sudo docker run hello-world**
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
7050e35b49f5: Pull complete
Digest: sha256:faa03e786c97f07ef34423fcccеeec2398ec8a5759259f94d99078f264e9d7af Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(arm64v8)
3. The Docker daemon created a new container from that image which runs the
executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
to your terminal.
To try something more ambitious, you can run an Ubuntu container with: $ docker run -it ubuntu bash
Share images, automate workflows, and more with a free Docker ID: https://hub.docker.com/
For more examples and ideas, visit: https://docs.docker.com/get-started/
3. Запустить контейнер с Ubuntu.
futitisme@futitismeserver:~$ **sudo docker search ubuntu**
NAME DESCRIPTION
ubuntu Ubuntu is a Debian-based Linux operating sys... 15321 [OK] futitisme@futitismeserver:~$ **sudo docker pull ubuntu**
Using default tag: latest
latest: Pulling from library/ubuntu
0509fae36eb0: Pull complete
Digest: sha256:4b1d0c4a2d2aaf63b37111f34eb9fa89fa1bf53dd6e4ca954d47caebca4005c2 Status: Downloaded newer image for ubuntu:latest

docker.io/library/ubuntu:latest
futitisme@futitismeserver:~$ **sudo docker images**

STARS OFFICIAL AUTOMATED

REPOSITORY TAG IMAGE ID
ubuntu latest 3c2df5585507
hello-world latest 46331d942d63 futitisme@futitismeserver:~$ **sudo docker run -it ubuntu**
root@a6238edccd96:/#

4. * Используя Dockerfile, собрать связку nginx + PHP-FPM в одном контейнере.

futitisme@futitismeserver:~$ **nano Dockerfile** FROM ubuntu:latest
MAINTAINER futitisme
RUN apt-get update
RUN apt-get install nginx
RUN apt-get install php-fpm
EXPOSE 80
CMD /user/sbin/nginx -g "daemon off:»
futitisme@futitismeserver:~$ **sudo docker build -t nginx-php-fpm .**
Successfully built 8f0c25a3f8e7
Successfully tagged nginx-php-fpm:latest
futitisme@futitismeserver:~$ **sudo docker run -d --name nginx-php-fpm -p 80:80 nginx-php-fpm:latest**
futitisme@futitismeserver:~$ **sudo docker ps -a**
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
8cc21c474443 nginx-php-fpm:latest "/bin/sh -c '/user/s..." 17 seconds ago Exited (127) 18 seconds ago nginx-php-fpm

Лабораторная 11. Доклад по утилитам OS Lunux.

https://github.com/Futitisme/OS_Linux/blob/main/os_linux_pdf.pdf

Выводы

В данной работе были рассмотрены утилиты MTR и MONITORIX, эти утилиты оказались крайне полезными, познакомился с синтаксисом и функциями, буду применять их в реальной разработке.