МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

На правах рукописи

Д.Л. Абдрахманов, Д.Н. Жариков,
А.Е. Андреев

Системное администрирование и DevOps

Учебное пособие

Волгоград
2021

УДК 004.45:004.056

Абдрахманов Д.Л.
Системное администрирование и DevOps: учеб. пособие / А.Е. Андреев, Д.Н. Жариков, Д.Л. Абдрахманов ВолгГТУ. – Волгоград, 2021. – 23 с.

В учебном пособии рассмотрены вопросы, связанные с администрированием вычислительных комплексов и центров обработки данных, а также особенности работы с ними.
Учебное пособие предназначено для магистров, обучающихся по программам магистратуры по профилю «искусственный интеллект» по направлениям 09.04.01 «Информатика и вычислительная техника», 09.04.03 «Прикладная информатика», 09.04.02 «Информационные системы и технологии». Учебное пособие выполнено в рамках реализации гранта на разработку программ бакалавриата и программ магистратуры по профилю «Искусственный интеллект», а также на повышение квалификации педагогических работников образовательных организаций высшего образования в сфере искусственного интеллекта (конкурс 2021-ИИ-01 от 10.06.2021).

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Вычислительные кластера и кластера хранения и обработки данных являются одной из основ современной интернет инфраструктуры, базой для размещения облачных сервисов, инструментом обработки и хранения больших структурированных и неструктурированных массивов данных.

В данном курсе рассматриваются как общие вопросы эксплуатации комплексов вычислительных машин, включая их программные компоненты и инженерную инфраструктуру, так и особенности их применения с точки зрения администратора системы.

1. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ К ПРАКТИЧЕСКИМ ЗАНЯТИЯМ

1.1. Практика №1. Введение в системное администрирование и DevOps.

1.1.1. Цель практической работы

Цель практической работы №1 состоит в том, чтобы на практических примерах дать студентам дать общее представление о программных компонентах ОС Linux. Особенностях архитектуры систем на базе данного ядра, топологии, уровнях надежности, а также о использовании системно доступной документации (мануалах).

1.1.2. Описание практической работы

Рассматриваются практические примеры основных программных компонентов Linux, их вызовы, особенности архитектуры, топологии, уровни ограничения доступа. Рассматриваются перспективы Linux.

1.2. Практика №2. Основы администрирования ОС Linux.

1.2.1. Цель практической работы

Цель практической работы №2 состоит в том, чтобы рассмотреть на практике структуру вычислительных систем, основные права доступа администратора и пользователя в файловой системе Linux.

1.2.2. Описание практической работы

Рассматриваются практические примеры применения программных продуктов ОС на базе ядра Linux в целях просмотра и модификации прав доступа пользователей к файлам в файловой системе.

Рассматриваются основные права доступа к файлу, права пользователя, пользователей входящих в группу и также пользователей не относящихся к

перечисленным ранее. Объясняются особенности работы с файлами в режиме суперпользователя.

1.3. Практика №3. Подходы к распределению задач в рамках единой среды исполнения. Использование прослоек абстракции от оборудования (HAL) при постановке задач. Особенности управления доступом пользователей к конкретным узлам в Slurm. Энергосбережение.

1.3.1. Цель практической работы

Цель практической работы №3 состоит в том, чтобы на практических примерах дать студентам общее представление о видах автоматического распределения задач в вычислительном комплексе.

1.3.2. Описание практической работы

Рассматриваются на практике основные виды программного обеспечения систем постановки задач в очередь PBS, Torque, Slurm, особенности настройки и использования.

1.4. Практика №4. Особенности настройки компонентов мониторинга Grafana. Использование программного обеспечения на языке python/bash для создания собственных метрик.

1.4.1. Цель практической работы

Цель практической работы №4 состоит в том, чтобы на практических примерах дать студентам общее представление о создании метрик мониторинга в системе Grafana.

1.4.2. Описание практической работы

Рассматриваются практические примеры задач и методов инженерии знаний, систем, основанных на знаниях. Дается обзор задач и методов систем бизнес-аналитики. Рассматриваются форматы представления

данных и инструментальные средства их обработки и преобразования. Рассматривается моделирование бизнес-аналитики с помощью нотации языка UML.

1.5. Практика №5. Использование Docker. Особенности применения в задачах и системах искусственного интеллекта, в том числе в задачах машинного обучения.

1.5.1. Цель практической работы

Цель практической работы № 5 состоит в том, чтобы на практических примерах дать студентам дать общее представление об особенностях применения Docker в задачах искусственного интеллекта и контейнеризации расчетов.

1.5.2. Описание практической работы

Рассматриваются особенности архитектур, подсистем, программного обеспечения и защиты информации для задач анализа данных, машинного обучения и систем, основанных на высоких требованиях к вычислительным мощностям. Особенности Docker для поддержки систем бизнес-аналитики, построение и масштабирование облачных ресурсов для задач ИИ. Рассматривается взаимодействие различных Docker контейнеров. Также даётся инструкция по развёртыванию отдельных образов Docker и их оркестрации через Docker Swarm/k8s.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

2.1 Лабораторная работа № 1. Знакомство с представлением данных получаемых из различных устройств в Linux.

2.1.1 Цели и задачи

Целью работы является ознакомление с представлением файлов в Linux общие принципы работы файловой системы, его составляющие, с приложениями осуществляющими чтение и запись для блоковых и символьных устройств.

Задачи :

1. Освоить базовые приложения консоли Linux,
2. Выбрать устройство для чтения данных,
3. Выполнить чтение данных из устройства,
4. Осуществить преобразование полученных данных в человеко читаемый формат.

2.1.2 Теоретические положения

Теоретические положения отражены в стандартах на проектирование и требования к вычислительным кластерам, включая отечественные стандарты ГОСТ, СНИП, требования СН 512-78, стандарты Tier Uptime Institute, а также в методических рекомендациях Минстроя РФ.

2.1.3 Порядок выполнения работы

1. Рассмотрение примера использования консольных программных компонентов по заданным входным данным, флагам и особенностям.
2. Выбор устройства по вариантам.

3. Рассмотрение примеров чтения/отправки данных через системные компоненты.

4. Получение данных из устройства.

5. Выполнение преобразования данных в понятный человеку вид.

6. Добавление возможности отслеживания в реальном времени

2.1.4. Варианты заданий

Считать данные из устройств по вариантам :

1. Получение данных из заданной консоли,

2. Получение текущих частот центрального процессора,

3. Получение текущего списка задач пользователя,

4. Получение данных с устройства, подключённого к COM порту.

Для каждого варианта задаются требования к базовому виду получаемых пользователем данных из реализованного приложения, что формирует множество вариантов.

2.1.5 Требования и состав отчёта

1. Отчёт должен быть выполнен на листах размера А4.

2. Отчёт должен начинаться с титульного листа с названием вуза и факультета, номером и названием лабораторной работы, вариантом, ФИО студента, № группы, ФИО преподавателя, городом и годом.

3. В отчёте нужно кратко описать задание, показать основные этапы решения задачи, сформулировать выводы.

4. Отчёт предоставить в бумажном или электронном виде (записать на флэш-накопитель и продублировать на электронную почту).

2.1.6 Вопросы и задания

1. Повторить и закрепить информацию из ГОСТ Р 58811-2020, 16325-88 и инструкции СН 512-78.
2. Повторить и закрепить информацию из стандарта Tier Uptime Institute.
3. Повторить и закрепить информацию из методических рекомендаций Министерства строительства РФ.
4. При защите отчёта надо уметь отвечать на вопросы по постановке задачи, этапам ее решения, использованным инструментам, формулам, справочникам и нормативным документам.

2.2 Лабораторная работа № 2 Постановка задач в Slurm, проверка ограничений системы.

2.2.1 Цели и задачи

Целью работы является ознакомление с общими принципами постановки задач в системах очередей Slurm.

Задачи :

1. Рассмотреть базовые команды постановки задач в очередь, просмотра состояния поставленных задач.
2. Выполнить постановку задачи в очередь, просмотр её состояния и также рассмотреть варианты приостановки и завершения выполнения задач.
3. Рассмотреть мониторинг активности пользователей и использования оборудования.

2.2.2 Теоретические положения

Теоретические положения отражены в стандартах на проектирование и требования к вычислительным комплексам, включая отечественные стандарты ГОСТ, СНИП, требования СН 512-78, стандарты Tier Uptime Institute, а также в методических рекомендациях Минстроя РФ.

Общие технические рекомендации по защите информации отражены в руководствах по администрированию ОС Linux.

2.2.2.1 Как пользоваться Slurm

Для того чтобы запустить задачу в очередь, нужно составить небольшой скрипт (назовём его job.sh) и выполнить команду sbatch job.sh

Посмотрим, что обычно находится внутри job.sh.

MPI

```
#!/bin/bash
# Название расчитываемой задачи. Может быть любым.
#SBATCH --job-name="sic"
#
# Множество вычислительных узлов для расчета задачи. Определяет характеристику
# вычислительных узлов.
#SBATCH --partition=intelv3-batch
#
# Запускать каждый расчет на одном узле.
#SBATCH --nodes=1
#
# Расчетное время, после истечения которого задача будет принудительно
# остановлена. В данном случае --- 7 дней.
#SBATCH --time=7-00:00:00
#
# Количество потоков одного процессора (20 для intelv3-batch, 24 для
# intelv4-batch, 256 для knl-batch).
#SBATCH --ntasks-per-node=20
# Чтобы srun заработал с impi
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so
srun pw.x < scf.in > scf.out
```

В данном варианте будет создано 20 MPI процессов Скрипт pw.x будет выполнен с входными данными из файла scf.in, после чего данные будут выведены в файл scf.out

GPU

Допустим, вы обучаете нейронные сети на питоне и вам захотелось использовать gpu.

```
#!/bin/bash
#
#SBATCH --job-name="Neural network"
#SBATCH --partition=intelv3-batch
srun --gres=gpu:1 -n1 -N1-1 --exclusive ./miniconda3/bin/python train.py
srun --gres=gpu:1 -n1 -N1-1 --exclusive ./miniconda3/bin/python classify.py
```

При запуске нужно явно указать, что тебе нужен gpu в опции –gres. Можно даже два попросить, если есть:

sbatch --gres=gpu:2 job.sh

2.2.2.2 Дополнительные примеры

Для кода, совмещающего в себе использование MPI и OpenMP, следует использовать следующую форму запускающего скрипта:

```
#!/bin/bash
# Название расчитываемой задачи. Может быть любым.
#SBATCH --job-name="sc2"
#
# Множество вычислительных узлов для расчета задачи. Определяет характеристику
# вычислительных узлов.
#SBATCH --partition=intelv4-batch
#
# Запускать расчет на одном узле.
#SBATCH --nodes=1
#
# Расчетное время, после истечения которого задача будет принудительно
# остановлена. В данном случае --- 7 дней.
#SBATCH --time=7-00:00:00
#
# Количество процессов одного узла
#SBATCH --ntasks-per-node=2
# Количество потоков создаваемое каждым процессом
#SBATCH --cpus-per-task=12
```

```
# Количество потоков соответствует перемножению количества процессов
# на количество созданных им потоков
# Количество потоков одного процессора (20 для intelv3-batch, 24 для
# intelv4-batch, 256 для knl-batch).
# Чтобы srun заработал с impi
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so
srun pw.x < scf.in > scf.out
```

В данном случае будет создано 24 потока вычислений на 1 узле. Можно также изменить количество узлов вычисления до определённого числа, заменив следующую опцию:

```
# Запускать расчет на нескольких узлах.
#SBATCH --nodes=2
```

А также определять конкретные узлы, на которых будет запускаться ПО, дописав:

```
# Запускать расчет на нескольких узлах.
#SBATCH --nodelist=node51.cluster,node52.cluster
```

2.2.3 Порядок выполнения работы

1. Постановка задачи в очередь.
2. Рассмотрение примеров успешной и неудачной постановки.
3. Подключение к рабочей в очереди задаче и её успешная приостановка с возобновлением работы.
4. Завершение выполнения задачи.

2.2.4. Варианты заданий

В качестве индивидуального варианта студенты рассматривают постановку различных системных и/или самописных приложений в очередь.

2.2.5 Требования и состав отчёта

1. Отчёт должен быть выполнен на листах размера А4.

2. Отчёт должен начинаться с титульного листа с названием вуза и факультета, номером и названием лабораторной работы, вариантом, ФИО студента, № группы, ФИО преподавателя, городом и годом.

3. В отчёте нужно кратко описать задание, показать основные этапы решения задачи, сформулировать выводы.

4. Отчёт предоставить в бумажном или электронном виде (записать на флэш-накопитель и продублировать на электронную почту).

2.2.6 Вопросы и задания

1. Повторить и закрепить информацию из ГОСТ Р 58811-2020, 16325-88 и инструкции СН 512-78.

2. Повторить и закрепить информацию из пособия «Защита информации в центрах обработки данных» И. А. Ушакова, В. А. Десницкого, А. А. Чечулина.

3. Повторить и закрепить информацию из курса по администрированию ОС Linux «Администрирование ОС Linux» [Электронный ресурс] – Режим доступа : https://intuit.ru/studies/courses/23/23/info.

2.3 Лабораторная работа № 3 Использование оркестратора Ansible для синхронизации состояния вычислительных узлов ЦОД. Использование Docker. Kubernetes.

2.3.1 Цели и задачи

Целью работы является ознакомление с особенностями работы оркестратора Ansible.

Задачи :

1. Рассмотреть задачи и инструменты Ansible в контексте высокопроизводительных многоузловых систем.

2. Рассмотреть особенности применения Ansible в задачах анализа данных.

3. Рассмотреть особенности применения Ansible в целях запуска задач.
4. Рассмотреть базовые методы использования контейнеров Docker.

2.3.2 Теоретические положения

Теоретические положения отражены в руководстве по использованию UML в системах бизнес-аналитики, в учебнике А.М. Блюмина «Проектирование систем интеллектуального обслуживания», в учебных пособиях по инженерии знаний из списка литературы данного пособия.

2.3.3 Порядок выполнения работы

1. Рассмотрение инструментов и методов Ansible.
2. Рассмотрение форматов команд и исходных файлов.
3. Рассмотрение особенностей применения Ansible, его структуры и компонент, программного обеспечения, включая инструменты для использования различных программных компонентов.
4. Использование базовых команд Docker для развёртывания базовых образов.

2.3.4. Варианты заданий

В качестве индивидуального варианта студенты рассматривают использование Ansible для системных операций, включая :

1. Множественное копирование файлов с переназначением прав доступа
2. Переименование файлов в директориях, создание мягких и жёстких ссылок
3. Использование системных приложений множества узлов с заметным эффектом.

2.3.5 Требования и состав отчёта

1. Отчёт должен быть выполнен на листах размера А4.

2. Отчёт должен начинаться с титульного листа с названием вуза и факультета, номером и названием лабораторной работы, вариантом, ФИО студента, № группы, ФИО преподавателя, городом и годом.

3. В отчёте нужно кратко описать задание, показать основные этапы решения задачи, сформулировать выводы.

4. Отчёт предоставить в бумажном или электронном виде (записать на флэш-накопитель и продублировать на электронную почту).

2.3.6 Вопросы и задания

1. Повторить и закрепить информацию из учебных пособий по инженерии знаний «Инженерия знаний. Модели и методы», «Инженерия знаний : учебное пособие», «Практикум по системам управления знаниями в организационно-экономических и производственно-технических системах» из списка литературы к дисциплине.

2. Повторить и закрепить информацию из учебника А.М. Блюмина «Проектирование систем интеллектуального обслуживания»

3. Повторить и закрепить информацию из пособия «Основы проектирования информационных систем с помощью языка UML».

2.4 Лабораторная работа № 4 Создание приложения для вывода информации о текущем состоянии оборудования на языке python

2.4.1 Цели и задачи

Целью работы является ознакомление с особенностями работы интерпретатора python.

Задачи :

1. Рассмотреть особенности применения python в задачах получения данных от системных приложений.

2. Рассмотреть особенности применения python в целях создания оповещений.

3. Рассмотреть базовые методы создания бесконечных циклов в целях постоянного получения данных.

4. Рассмотреть задачи и инструменты screen в контексте поддержания работоспособности приложений.

2.4.2 Теоретические положения

Теоретические положения отражены в руководстве по использованию UML в системах бизнес-аналитики, в учебнике А.М. Блюмина «Проектирование систем интеллектуального обслуживания», в учебных пособиях по инженерии знаний из списка литературы данного пособия.

2.4.2.1 Отправка простого оповещения

Откроем системный терминал при помощи комбинации alt+f2 введя в открывшемся окне gnome-terminal

Попробуем использовать python для создания простейших оповещений.

Проверим наличие python3 на компьютере

```
$ python3
```

Если среда запустилась и отобразила версию вводим следующее:

```
>>> import notify2
```

В результате получим информацию о том, что данный модуль отсутствует.

Для установки модуля выходим из интерпретатора python сочетание клавиш Ctrl+D и запускаем установщик модулей python

```
$ pip3 install notify2
```

Но система скорее всего скажет о том что установщика модулей тоже нет, однако сразу же предложит решение:

```
$ sudo apt install python3-pip
```

После успешной установки снова вводим

```
$ pip3 install notify2
```

И можно снова войти в python3 и попробовать снова загрузить модуль

```
$ python3
```

```
>>> import notify2
```

Теперь не выходя из интерпретатора делаем следующее

```
>>> notify2.init("Test")
>>> notice = notify2.Notification(‘Hello’, ‘World’)
>>> notice.show()
```

После этого вы увидите системное оповещение.

Теперь можно выйти из интерпретатора python через сочетание клавиш Ctrl+D

2.4.2.2 Работа через файл

Давайте создадим файл для нашей программы

Выполним в терминале

```
$ cd
$ mkdir notify
$ cd notify/
$ touch notify
```

Только что мы перешли в домашнюю директорию и создали в ней папку notify и перейдя в ней создали файл notify

Давайте попробуем что-нибудь записать в файл

```
$ nano notify
```

И в открывшееся окно запишем:

```
import notify2
notify2.init("Test")
notice=notify2.Notification('Hello', 'World')
notice.show()
```

После чего сохраним через Ctrl+O Enter

И выйдем через Ctrl+X

Теперь можно вызывать наше оповещение, используя команду:

```
$ python3 notify
```

Сейчас наш python код просто посылает сообщение с заданным заранее содержанием. Но давайте попробуем послать что-то более интересное:

1 этап - выделим функцию и вызовем её:

```
$ nano notify
```

И заполним

```
import notify2
def send_message(header, message):
    notify2.init("Test")
    notice=notify2.Notification(header, message)
    notice.show()
send_message("Hello there", "Kenobi")
```

Правило — если для строк внутри функции send_message используется отступ по tab то внутри приложения всегда должен использоваться tab.

Снова сохраним (Ctrl+O Ctrl+X) и вызовем.

```
$ python3 notify
```

2 этап — кастомные данные:

Итак, для того чтобы о чём-то оповещать надо что-то отслеживать

```
$ nano notify
```

Импортируем новый модуль

```
import notify2
import subprocess
def send_message(header, message):
    notify2.init("Test")
    notice=notify2.Notification(header, message)
    notice.show()
c=subprocess.check_output('/usr/bin/lscpu', shell=True) # Узнаём данные о процессоре
send_message("CPU state", c)
```

И запустив приложение сейчас пусть и видим интересные данные, но понимаем что нотификация должна быть более лаконичной короткой и ясной.

После всего вышеописанного наш код представляет из себя следующее:

```
import notify2
import subprocess
def send_message(header, message):
    notify2.init("Test")
    notice=notify2.Notification(header, message)
    notice.show()
c = subprocess.check_output('/usr/bin/lscpu', shell=True)
code = c.decode("utf-8")
data_array = code.split('\n')
for i in data_array:
    mini=i.split(':')
    if mini[0]=="CPU MHz":
        bgm=mini[1].replace(' ',")
        send_message(mini[0], bgm)
```

На этой стадии наше оповещение уже работает, но давайте немного усложним и сделаем более красиво.

2.4.2.3 Изменяем формат

Итак, у нас есть частота на экране, но стоит сделать более красивым её формат.

Для этого ещё немного поработаем со строками

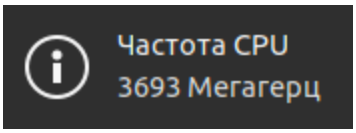
Оставляем от 1 строки только слово CPU и добавляем понятное нам слово частота

```
cpu=mini[0].split(' ')
title="Частота " + cpu[0]
```

Оставляем только целое значение Мгц

```
foot=bgm.split('.')
message=foot[0]+" Мегагерц"
```

После преобразований получаем



Радуемся понятной русской речи.

2.4.2.4 Итоговое содержимое файла нотификации

```
import notify2
import subprocess
def send_message(header, message):
    notify2.init("Test")
    notice=notify2.Notification(header, message)
    notice.show()
c = subprocess.check_output('/usr/bin/lscpu', shell=True)
code = c.decode("utf-8")
data_array = code.split('\n')
for i in data_array:
    mini=i.split(':')
    if mini[0]=="CPU MHz":
        cpu=mini[0].split(' ')
        title="Частота " + cpu[0]
        bgm=mini[1].replace(' ',")
        foot=bgm.split('.')
        message=foot[0]+" Мегагерц"
        send_message(title, message)
```

2.4.2.5 Повторные оповещения

Во многих языках программирования есть конструкция ***while*** которая выполняет тело цикла до тех пор, пока условие истинно.

Истинность условия подразумевает результат выражения равный ***true***.

Соответственно ***true*** - всегда равен ***true***, и таким образом ***while(true){}*** бесконечный цикл

Потому код с добавлением всего 1 строки

```
import notify2
import subprocess
def send_message(header, message):
    notify2.init("Test")
    notice=notify2.Notification(header, message)
    notice.show()
while(1):
    c = subprocess.check_output('/usr/bin/lscpu', shell=True)
    code = c.decode("utf-8")
    data_array = code.split('\n')
    for i in data_array:
        mini=i.split(':')
        if mini[0]=="CPU MHz":
            cpu=mini[0].split(' ')
            title="Частота " + cpu[0]
            bgm=mini[1].replace(' ',")
            foot=bgm.split('.')
            message=foot[0]+" Мегагерц"
            send_message(title, message)
```

Но теперь проблема состоит в том что оповещения приходят постоянно. Время создать условие, по которому мы хотим их получать.

2.4.2.6 Добавление задержек

Итак, нам не нужны частые оповещения, нам надо знать когда ситуация изменилась или о ситуациях когда наблюдаемое явление не меняется, а значит делаем следующее:

Необходимо, чтобы у нас была возможность не нагружать компьютер постоянно нашими запросами while(true), а создать ситуацию при которой мы запрашиваем данные по частоте хотя бы раз в секунду.

Для этого нам понадобится задержка, а её можно найти в модуле time

```
import time
time.sleep(1)
```

А заодно давайте проверять такие вещи как частота, которая уже была и сравнивать её с текущей

```
if int(foot[0])!=freq:
freq=int(foot[0])
counter=0
send_message(title, message)
```

И заодно отдавать данные каждые 60 секунд, если частота не менялась

```
if counter==60:
counter=0
title="Частота не изменилась"
send_message(title, message)
```

После всего прибавляем значение таймера

```
counter=counter+1
```

2.4.2.7 Итоговый код

```
import notify2
import subprocess
import time
def send_message(header, message):
```

```
        notify2.init("Test")
        notice=notify2.Notification(header, message)
        notice.show()
freq=0
counter=60
while(1):
        c = subprocess.check_output('/usr/bin/lscpu', shell=True)
        code = c.decode("utf-8")
        data_array = code.split('\n')
        for i in data_array:
                mini=i.split(':')
                if mini[0]=="CPU MHz":
                        cpu=mini[0].split(' ')
                        title="Частота " + cpu[0]
                        bgm=mini[1].replace(' ',")
                        foot=bgm.split('.')
                        message=foot[0]+" Мегагерц"
                        time.sleep(1)
                        if int(foot[0])!=freq:
                                freq=int(foot[0])
                                counter=0
                                send_message(title, message)
                        if counter==60:
                                counter=0
                                title="Частота не изменилась"
                                send_message(title, message)
                        counter=counter+1
```

Но мы всё ещё привязаны к консоли из которой запускаем. Если консоль закрыть, то выполнение завершится. Для этого есть пакет screen. Попробуем запустить его.

```
$ screen
```

Если не получится — установим

```
$ sudo apt install screen
```

Зайдём в него

```
$ screen
```

И запустим наш скрипт

```
$ python3 notify
```

И теперь можно выйти по Ctrl+A затем D и ваше приложение нотификаций продолжит работать

2.4.3 Порядок выполнения работы

1. Рассмотрение инструментов и методов python.
2. Рассмотрение форматов команд и исходных файлов.
3. Рассмотрение особенностей применения python, его структуры и компонент, программного обеспечения, включая инструменты для использования различных программных компонентов.
4. Использование базовых команд python для развёртывания базовых приложений.

2.4.4. Варианты заданий

В качестве индивидуального варианта студенты рассматривают использование python для системных операций, включая :

1. Получение данных списка активных pci express устройств
2. Получение данных о текуших активных процессах пользователей
3. Получение информации о температурах оборудования.

2.4.5 Требования и состав отчёта

1. Отчёт должен быть выполнен на листах размера А4.

2. Отчёт должен начинаться с титульного листа с названием вуза и факультета, номером и названием лабораторной работы, вариантом, ФИО студента, № группы, ФИО преподавателя, городом и годом.

3. В отчёте нужно кратко описать задание, показать основные этапы решения задачи, сформулировать выводы.

4. Отчёт предоставить в бумажном или электронном виде (записать на флэш-накопитель и продублировать на электронную почту).

2.4.6 Вопросы и задания

1. Повторить и закрепить информацию из учебных пособий по инженерии знаний «Инженерия знаний. Модели и методы», «Инженерия знаний : учебное пособие», «Практикум по системам управления знаниями в организационно-экономических и производственно-технических системах» из списка литературы к дисциплине.

2. Повторить и закрепить информацию из учебника А.М. Блюмина «Проектирование систем интеллектуального обслуживания»

3. Повторить и закрепить информацию из пособия «Основы проектирования информационных систем с помощью языка UML».

3. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

3.1. Задание на контрольную работу и методические указания по ее выполнению

На контрольную работу студенту выдается индивидуальное задание (по вариантам), заключающееся в начальных этапах работы системного администратора в ОС семейства Linux.

Работа выполняется параллельно и в контексте индивидуальных заданий к лабораторному практикуму по дисциплине. Оформляется в письменной форме в течение 10 недель с момента выдачи задания. Контрольный срок сдачи – последний месяц семестра.

Правила оформления контрольной работы

- контрольная работа оформляется в редакторе MS Word / OpenOffice (*.doc, *.docx, *.odt);
- листы формата А4, ориентация книжная;
- поля: левое – 2 см, остальные – по 1 см;
- шрифт – Times New Roman;
- размер шрифта 14 pt;
- междустрочный интервал – 1,5;
- абзацный отступ – 1,25 см;
- нумерация страниц сквозная, номер на первой странице не ставится;
- в конце работы необходим список использованной литературы согласно ГОСТ Р 7.0.5 – 2008;
- объем работы зависит от степени раскрытия основных пунктов контрольной работы.

3.2. Примерное содержание контрольной работы

Примерное содержание контрольной работы

1.Титульный лист.

2.Формулировка варианта задания.

3.Основная часть, включающая:

1) описание вычислительных систем;

2) разработка и описание требований к конфигурации оборудования;

3) разработка и описание требований к конфигурации электропитания и охлаждения;

4) разработка и описание требований к программному обеспечению;

5) разработка и описание требований к администрированию и защите информации;

6) описание выбора программных компонентов вычислительного кластера и проектных решений;

7) описание произведенных настроек и расчетов;

8) краткое описание основных настроек программного обеспечения.

6) Список использованных источников (включая источники Интернет).

3.3. Примерные варианты заданий контрольной работы

Примерный список вариантов контрольной работы :

1. Начальное проектирование вычислительного комплекса для хранения и обработки больших данных в области промышленности
2. Начальное проектирование вычислительного комплекса для хранения и обработки больших данных в области медицины
3. Начальное проектирование вычислительного комплекса для хранения и обработки больших данных в области видеонаблюдения
4. Начальное проектирование вычислительного комплекса для хранения и обработки больших объемов речевых данных

5. Начальное проектирование вычислительного комплекса для хранения и обработки больших объемов данных сенсорной сети IoT

6. Начальное проектирование вычислительного комплекса для поддержки системы, основанной на знаниях.

7. Модернизация вычислительного комплекса для хранения и обработки больших объемов данных сенсорной сети IoT

8. Модернизация вычислительного комплекса для хранения и обработки больших объемов метеорологических (медицинских, промышленных и др.) данных.

9. Модернизация вычислительного комплекса для поддержки системы основанной на знаниях.

10. Модернизация вычислительного комплекса для поддержки задач бизнес-аналитики.

ЗАКЛЮЧЕНИЕ

В рамках курса на практических примерах и в лабораторном практикуме рассматриваются общие вопросы администрирования и эксплуатации вычислительных комплексов, включая их компоненты и инженерную инфраструктуру, нормативная база для обслуживания кластеров, а также особенности применения Slurm для поддержки высокопроизводительных систем на примере задач из области обработки больших данных.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА ПО КУРСУ

1. Сейерс, Э. Х. Docker на практике / Э. Х. Сейерс, А. Милл ; перевод с английского Д. А. Беликов. — Москва : ДМК Пресс, 2020. — 516 с. — ISBN 978-5-97060-772-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131719 (дата обращения: 12.10.2021).

2. Защита информации в центрах обработки данных : учебное пособие / И. А. Ушаков, В. А. Десницкий, А. А. Чечулин [и др.]. — Санкт-Петербург : СПбГУТ им. М.А. Бонч-Бруевича, 2019. — 92 с. — Текст : электронный // Лань : электронно-библиотечная система. — Режим доступа: https://e.lanbook.com/book/180085 (дата обращения: 10.10.2021).

3. Администрирование ОС Linux [Электронный ресурс] – Режим доступа : https://intuit.ru/studies/courses/23/23/info

4. Операционная система Linux [Электронный ресурс] – Режим доступа : https://intuit.ru/studies/courses/37/37/info

5. Староверова, Н. А. Операционные системы : учебник / Н. А. Староверова. — Санкт-Петербург : Лань, 2019. — 308 с. — ISBN 978-5-8114-4000-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/125737

6. Ларина, Т. Б. Администрирование операционных систем. Управление системой : учебное пособие / Т. Б. Ларина. — Москва : РУТ (МИИТ), 2020. — 71 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/175980

7. Романов, С. Л. Работа в операционной среде Linux: практикум для вузов : учебное пособие / С. Л. Романов. — Санкт-Петербург : БГТУ "Военмех" им. Д.Ф. Устинова, 2017. — 74 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/121866

8. Введение в Linux : учебно-методическое пособие / составители М. А. Артемов [и др.]. — Воронеж : ВГУ, 2016. — 44 с. — Текст : электронный //

Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/165430

9. Лукша, М. Kubernetes в действии / М. Лукша ; перевод с английского А. В. Логунов. — Москва : ДМК Пресс, 2019. — 672 с. — ISBN 978-5-97060-657-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131688

Учебное издание

Дмитрий Леватович Абдрахманов
Дмитрий Николаевич Жариков
Андрей Евгеньевич Андреев

СИСТЕМНОЕ АДМИНИСТРИРОВАНИЕ И DEVOPS

Учебное пособие

Волгоградский государственный технический университет.
400005, г. Волгоград, просп. В. И. Ленина, 28, корп. 1.