Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра ИКСС

Контрольная работа

по дисциплине: «Операционные системы»

Вариант 7

Выполнил: студент

группы *NN*

*ФИО*

Проверил:

*ФИО*

**Теория:**

Под архитектурой операционной системы понимают структурную и *функциональную организацию* ОС на основе некоторой совокупности программных модулей. В состав ОС входят исполняемые и объектные модули стандартных для данной ОС форматов, программные модули специального формата (например, *загрузчик* ОС, *драйверы ввода-вывода*), конфигурационные файлы, файлы документации, модули справочной системы и т.д.

Операционная система состоит из четырех основных частей:

1. **Первая часть**— ядро, низкоуровневая основа любой ОС, выполняемая аппаратурой в особом **привилегированном режиме**. Ядро загружается в память один раз и находится в памяти **резидентно**– постоянно, по одним и тем же адресам. Ядро — командный интерпретатор, «переводчик» с программного языка на «железный», язык машинных кодов.
2. **Вторая часть**— **Подсистема управления ресурсами (resource allocator)** — управляет вычислительными ресурсами компьютера — оперативной и внешней памятью, процессором
3. **Третья часть — Управляющая программа (control program, supervisor) –**управляет исполнением других программ и функционированием устройств ввода-вывода.(используются специализированные программы для управления различными устройствами, входящими в состав компьютера. Драйвера «системные библиотеки»)
4. **Четвертая часть** — удобная оболочка, с которой общается пользователь — интерфейс. Своего рода красивая обертка, в которую упаковано скучное и не интересное для пользователя ядро.

Наиболее общим подходом к структуризации ОС является разделение всех ее модулей на две группы:

– ядро – модули, выполняющие основные функции ОС (управление памятью, устройствами ввода-вывода и т. д.);

– модули, выполняющие вспомогательные функции ОС.

Без ядра ОС является полностью неработоспособной и не может выполнить ни одну из своих функций. Крах ядра равносилен краху всей системы.

В состав ядра входят:

– функции, решающие внутрисистемные задачи организации вычислительного процесса (такие функции недоступны для приложений);

– функции для поддержки приложений. Эти функции создают для приложений так называемую прикладную программную среду;

– функции, которые могут вызываться приложениями, так как приложения могут обращаться к ядру с запросами (системными вызовами) для выполнения каких-либо действий, например для открытия и чтения файла, вывода графической информации на дисплей и т. д.

Обычно ядро оформляется в виде программного модуля некоторого специального формата, отличающегося от формата пользовательских приложений.

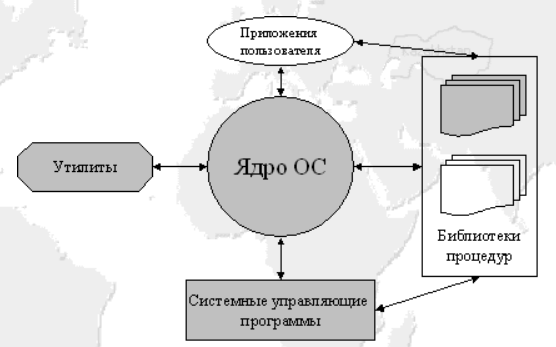
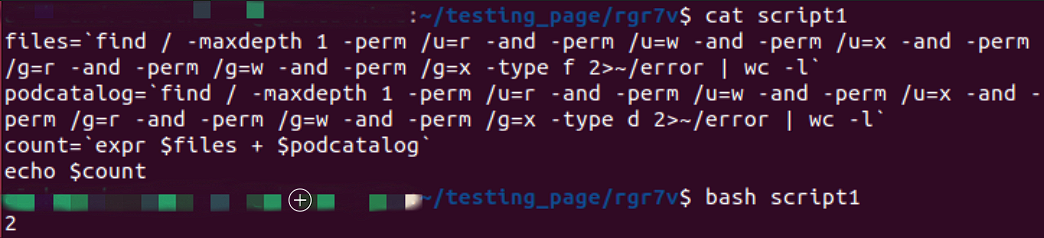


Рис. 1 - Взаимодействие между ядром и вспомогательными модулями ОС

**Источники:**

1. *Intuit «Лекция 1: Архитектура, назначение и функции операционных систем»* [электронный ресурс] // <https://intuit.ru/studies/courses/631/487/lecture/11048?page=4>
2. *Komputercnulja «Архитектура Операционной системы»* [электронный ресурс] // [komputercnulja.ru/operacionnye-sistemy/arxitektura-operacionnoj-sistemy](http://komputercnulja.ru/operacionnye-sistemy/arxitektura-operacionnoj-sistemy)
3. *Southsidebachelor «Операционные системы»* [электронный ресурс] // <https://southsidebachelor.wixsite.com/ossystems/arhitektura-operacionnoj-sistemy>
4. *Bourabai «Операционные системы вычислительных машин»* [электронный ресурс] // <http://bourabai.ru/os/lecture03.htm>

Задание 1.



files=`find / -maxdepth 1 -perm /u=r -and -perm /u=w -and -perm /u=x -and -perm /g=r -and -perm /g=w -and -perm /g=x -type f 2>~/error | wc -l`

# -maxdepth – максимальная глубина поиска, -perm – поиск по правам (r – чтение, w – запись, x – исполнение), u – для пользователя, g – для группы, -and – добавить ещё одно условие, wc -l – посчитать количество строчек в выводе команды find

podcatalog=`find / -maxdepth 1 -perm /u=r -and -perm /u=w -and -perm /u=x -and -perm /g=r -and -perm /g=w -and -perm /g=x -type d 2>~/error | wc -l`

# -maxdepth – максимальная глубина поиска, -perm – поиск по правам (r – чтение, w – запись, x – исполнение), u – для пользователя, g – для группы, -and – добавить ещё одно условие, wc -l – посчитать количество строчек в выводе команды find

count=`expr $files + $podcatalog`

# expr – команда (программа) для вычисления выражения, в данном случае сумма файлов и подкаталогов из результата выше

echo $count # вывод переменной count

Задание 2.



namesinterface=`ifconfig | cut -d" " -f1` # команда ifconfig выводит информацию о сетевых интерфейсах, cut -d “ “ -f1 – вырезает всё до первого пробела в выводе команды ifconfig

count=`echo $namesinterface | wc -l` # выводятся значения в переменной namesinterface и считается их количество

if [ $count -eq 0 ] # если количество интерфейсов равно 0

then # то

echo "Сетевых интерфейсов нет."

else # иначе

for i in $namesinterface # цикл по значениям в переменной namesinterface

do

IP=`ip -4 addr show $i | grep -oP '(?<=inet\s)\d+(\.\d+){3}'`

# пояснение про grep: -P(язык программирования Perl), ?<="слово"\s определить начало в точке "слово" и рассматривать слово справа после данного шаблона и пробела (только 1 строка!)

# например строчка: priv ky mir и если написать (?<=priv\s)\w+, то выведется строчка ky

# однако, если дописать в конце (\"разделитесь слов"\w+{"количество повторений вхождения разделителя"}, в нашем случае будет пробел и 1 раз, то выдаст: ky mir)

# d+("\"разделитель"\d+) - вывод числовых символов разделённых разделителем

# {"число повторений"} - количество повторений для вывода определённого количества слов разделённых определённым разделителем

# + ставится для того, чтобы рассматривались все символы, а не только один

# d+ - цифра | w+ - алфавитно-цифровой символ

echo -e "\033[01;33mInterface\033[00;0m: $i\n\033[01;38mIPv4 address\033[00;0m: $IP" # вывод названия интерфейса и его IPv4 address

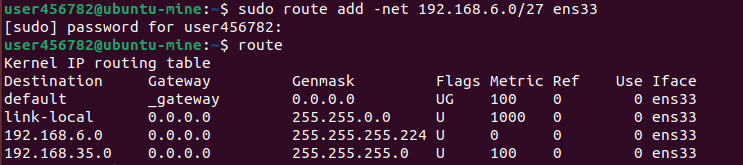
done

fi

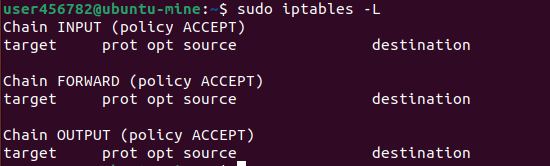
Задание 3.

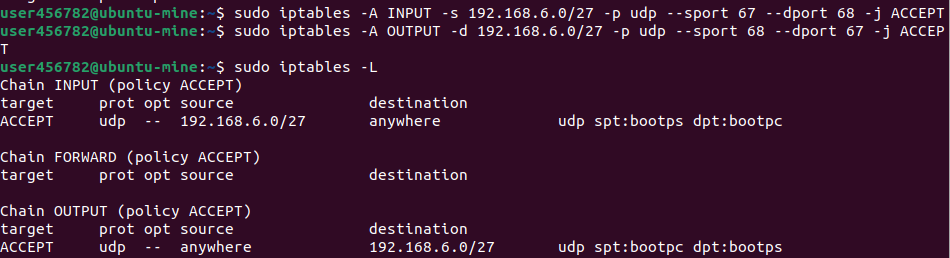
* 1. route add -net 192.168.6.0/27 ens33

Изображение выглядит как текст

Автоматически созданное описание  


2.

iptables -A INPUT -s 172.30.3.0/27 -p udp --sport 67 –dport 68 -j ACCEPT  
iptables -A OUTPUT -d 172.3.0/27 -p udp –sport 68 –dport 67 -j ACCEPT  


  
input – входящие (-s) dhcp запросы от хостов в подсети к которой настроена маршрутизация (172.30.3.0/27) -p протокол udp порт источника (--sport 67) порт отправителя (--dport 68) -j ACCEPT (если правило подошло)  
output – исходящие (-d) dhcp запросы от хостов в подсети к которой настроена маршрутизация (172.30.3.0/27) -p протокол udp порт источника (--sport 68) порт отправителя (--dport 67) -j ACCEPT (если правило подошло)