**Лабораторная работа №6. Продвинутая работа с системой**

к сведению

Основная задача: научиться управлять процессами и создавать простейшую автоматизацию.

**Автоматизируем сбор данных о системе с bash**

1. Выведем LA:

cat /proc/loadavg | awk '{ print $1,$2,$3" processes: "$4", last PID: "$5}'

**LA, Load average** - среднее значение загрузки системы за некоторый период времени, как правило, отображается в виде трёх значений, которые представляют собой усредненные сглаженные величины за последние 1, 5 и 15 минут. Вычисляется как длина очереди выполнения в операционной системе, где единица означает, что очередь заполнена, а значение выше единицы — что есть процессы, которые ожидают своей очереди на выполнение. Содержатся в файле /proc/loadavg.

Учитывает как running процессы, так и процессы в uninterraprable sleep (ожидают выполнения системного вызова, как правило долгие это read/write, т.е. IO).

1. Выведем список сетевых интерфейсов в нужном формате

ss -4tuln | подставьте что нужно, можно цепочку из нескольких команд | column -t

127.0.0.53%lo 53  
127.0.0.53%lo 53  
0.0.0.0 22

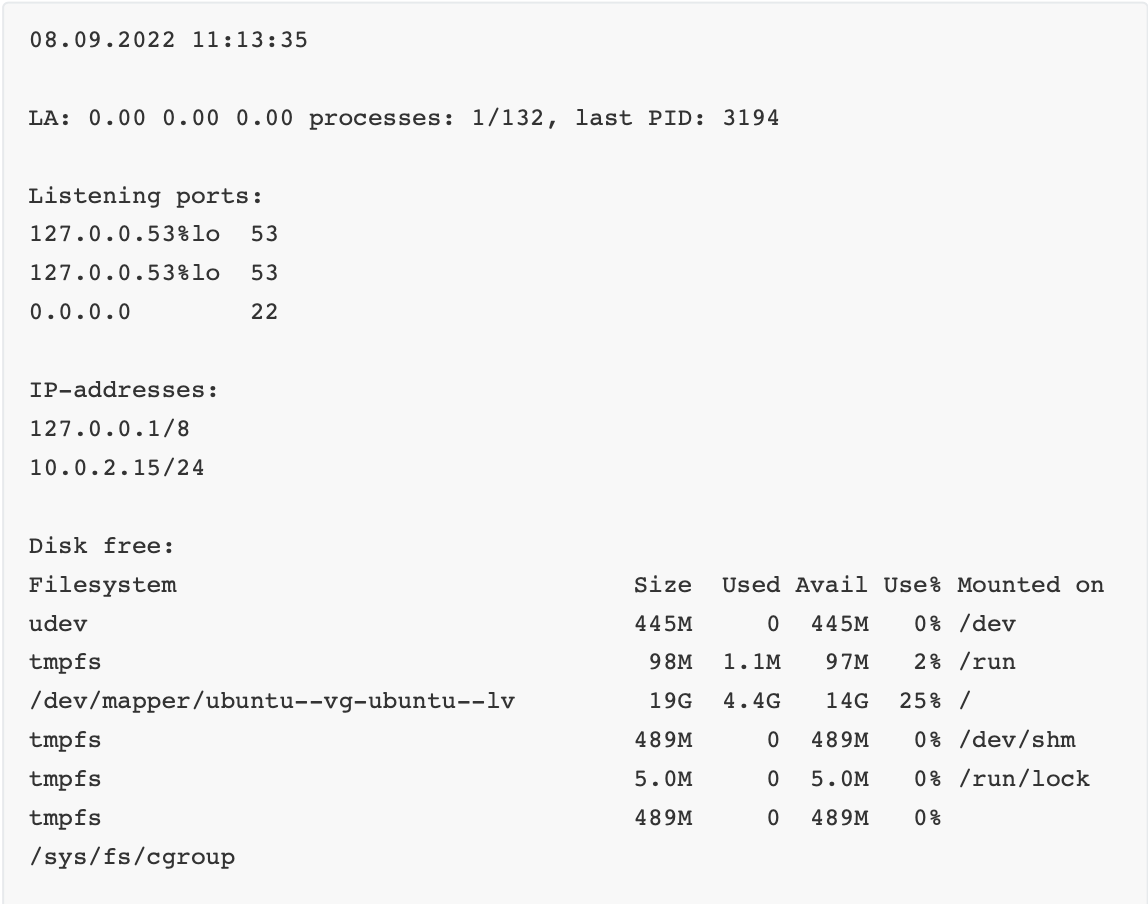
1. Выведем список IP-адресов на хосте:

ip a | grep -E 'inet ' | awk '{ print $2 }'

1. Обернем это все в bash-скрипт /root/beholder.sh, добавим date + "%d.%m.%Y %H:%M:%S" и информацию о занятом месте и списке пользователей (самостоятельно). Попробуем циклы, запишем в скрипт:

*for* i *in* $@  
*do*  
 echo "Next search by name:"  
 ps aux | grep $i  
 echo  
*done*

1. Запустим. Должны получить что-то вроде:



**Делаем сбор данных регулярным с cron**

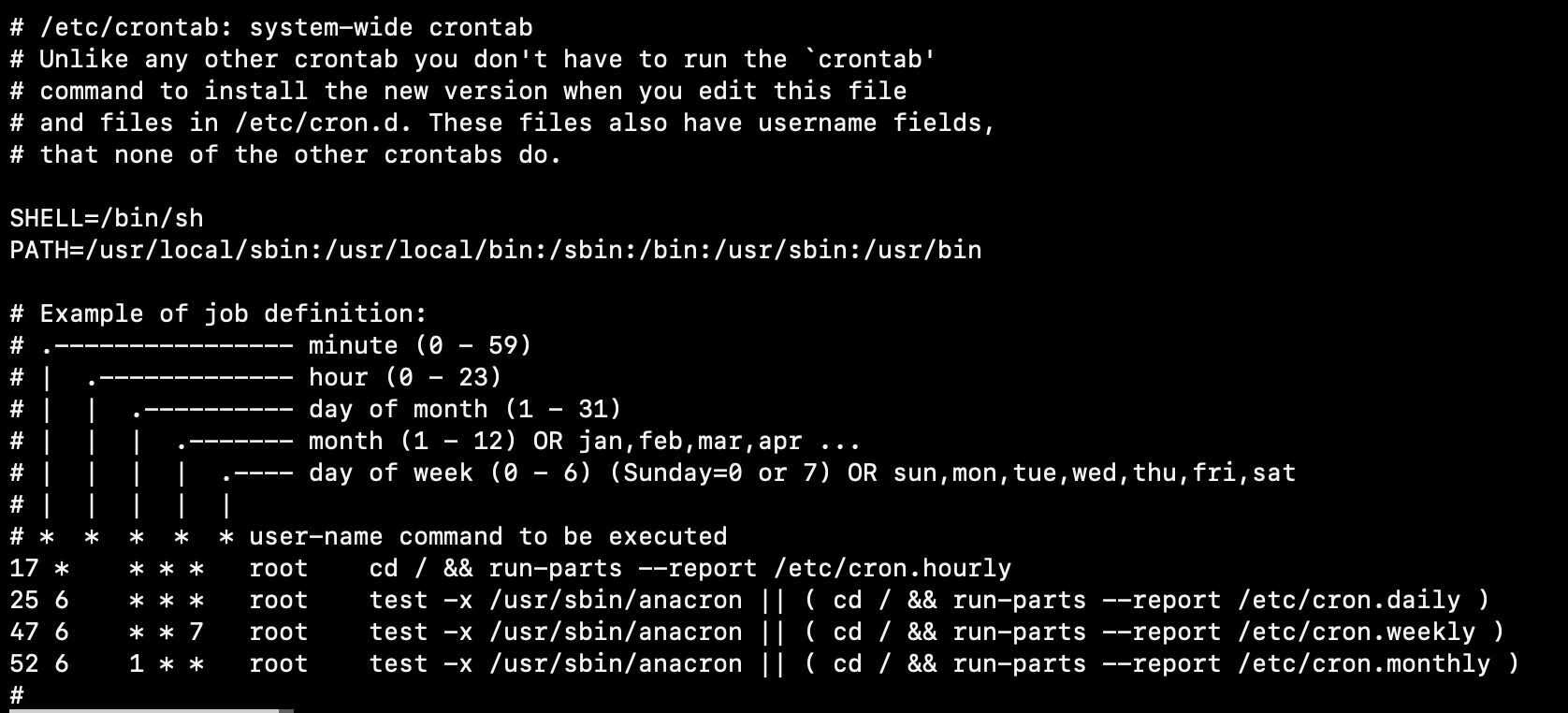
1. Впишем в cron новое правило и ждем минуту.

echo '\* \* \* \* \* root /root/beholder.sh >> /tmp/beholder-output' >  
/etc/cron.d/beholder

1. Смотрим файл cat /tmp/beholder-output

Можем попробовать поменять параметры, сервис [crontab.guru](https://crontab.guru/) в помощь.

Краткое описание есть в файле конфига /etc/crontab:



Исходя из этого конфига следует, что в /etc/cron.{hourly,daily, weekly,monthly}/ лежат скрипты, запускающиеся соответственно раз в день, в неделю и в месяц. Дополнительно Cron читает конфиги из /etc/cron.d/, где лежат общие файлы с расписанием запуска. Кроме того, для пользователей задаются персональные cron-конфиги. Их можно отредактировать с помощью утилиты crontab -e. Они отличаются от /etc/crontab тем, что там опускается поле пользователя, запускаться будет от того, который себе задал это в настройках.

**Пишем свой systemd-сервис**

Процесс не может взяться из ниоткуда: его обязательно должен запустить какой-то процесс. Процесс, запущенный другим процессом, называется дочерним (child) процессом или потомком. Процесс, который запустил процесс называется родительским (parent), родителем или просто - предком. У каждого процесса есть два атрибута - **PID** (Process ID) - идентификатор процесса и **PPID** (Parent Process ID) - идентификатор родительского процесса.

Процессы создают иерархию в виде дерева. Самым "главным" предком, то есть процессом, стоящим на вершине этого дерева, является процесс systemd или init (PID = 1). Условно можно считать, что планировщик ядра имеет PID = 0.

Собственно для управления древом процессов и создали systemd

[systemd](https://ru.wikipedia.org/wiki/Systemd) — подсистема инициализации и управления службами в GNU/Linux, фактически вытеснившая в 2010-е годы традиционную подсистему init. Основная особенность — интенсивное распараллеливание запуска служб в процессе загрузки системы, что позволяет существенно ускорить запуск операционной системы. Основная единица управления — unit (модуль), одним из типов модулей являются "сервисы" — аналог демонов — наборы процессов, запускаемые и управляемые средствами подсистемы и изолируемые контрольными группами.

Создадим свой systemd-unit:

nano /etc/systemd/system/myhttp.service  
  
*# Создадим для него WorkingDirectory*  
mkdir -p /root/www  
  
*# Перезагрузим конфигурацию*  
systemd systemctl daemon-reload

Содержимое .service файла:

[Unit]  
Description=MyHTTP Server  
Documentation=http://example.org/  
After=network.target  
  
[Service]  
WorkingDirectory=/root/www  
ExecStart=/usr/bin/python3 -m http.server 8000  
KillMode=mixed  
Restart=on-failure  
Type=simple  
  
[Install]  
WantedBy=multi-user.target  
Alias=myhttpd.service

Кратко разберем что тут написано:

* Секция [Unit]:
  + Description - описание, отображается в systemctl status [service]
  + Documentation - ссылка на документацию, обязательно в виде URL
  + After - после каких целей/сервисов следует запускать данный unit
* Секция [Service]:
  + WorkingDirectory - вызовет системный вызов chdir перед исполнением каких-либо команд сервиса, должна существовать
  + ExecStart - команда, которая запустится при вызове systemctl start [service]
  + KillMode - управляет тем, как процесс должен быть завершен, mixed - посылает SIGTERM основному процессу и SIGKILL дочерним
  + Restart - необходимо ли перезапускатьсервис в случае завершения процесса (в примере - только при ненулевом rc)
  + Type - настраивает процедуру запуска и отслеживания процесса, simple ожидает, что основным станет процесс, запущенный ExecStart командой и сервис будет работать до тех пор, пока жив основной процесс.
* Секция [Install]:
  + WantedBy - какая цель триггерит запуск, если сервис enabled.
  + Alias - альтернативное имя сервиса. systemctl status [alias] равнозначен systemctl start [service].

Запустим, добавим в автозапуск:

systemctl start myhttp  
systemctl enable myhttp

Проверяем работу с помощью curl и ss.

curl http://127.0.0.1:8000/  
ss -tl | grep 8000

**Запускаем процесс внутри нового пространства имен**

Посмотрим список доступных пространств имен:

lsns  
ip netns

Создаем сетевой namespace:

ip netns add myhttp

Теперь запустим внутри пару команд ip netns exec <netns> <cmd> ...:

ip netns exec myhttp ip link set dev lo up  
ip netns exec myhttp /usr/bin/python3 -m http.server 8080 &

Проверим:

ss -tul4n  
ip netns exec myhttp ss -tul4n

Запустим tcpdump внутри netns-а:

ip netns exec myhttp tcpdump -i any host 127.0.0.1

Проверим:

ip netns exec myhttp curl http://127.0.0.1:8080  
ip netns exec myhttp curl http://127.0.0.1:8000  
curl http://127.0.0.1:8080  
curl http://127.0.0.1:8000

Таким образом, у нас получилось разнести 2 разных сервера в разные сетевые пространства имен. Мы даже можем запустить их на одинаковом порту.

**Установка docker и запуск hello-world**

sudo apt install apt-transport-https ca-certificates curl software-properties-  
common  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
sudo add-apt-repository "deb [arch=amd64]  
https://download.docker.com/linux/ubuntu focal stable"  
sudo apt update  
sudo apt install docker-ce  
sudo systemctl status docker

Запустим hello-world:

docker run hello-world