1. Какой системный вызов делает команда cd? В прошлом ДЗ мы выяснили, что cd не является самостоятельной программой, это shell builtin, поэтому запустить strace непосредственно на cd не получится. Тем не менее, вы можете запустить strace на /bin/bash -c 'cd /tmp'. В этом случае вы увидите полный список системных вызовов, которые делает сам bash при старте. Вам нужно найти тот единственный, который относится именно к cd.

На сколько я понял вывод команды strace:

получает информацию о файле - stat("/tmp", {st\_mode=S\_IFDIR|S\_ISVTX|0777, st\_size=4096, ...}) = 0

изменяет директорию - chdir("/tmp")

2. Попробуйте использовать команду file на объекты разных типов на файловой системе. Например:

openat(AT\_FDCWD, "/usr/share/misc/magic.mgc"

3. Предположим, приложение пишет лог в текстовый файл. Этот файл оказался удален (deleted в lsof), однако возможности сигналом сказать приложению переоткрыть файлы или просто перезапустить приложение – нет. Так как приложение продолжает писать в удаленный файл, место на диске постепенно заканчивается. Основываясь на знаниях о перенаправлении потоков предложите способ обнуления открытого удаленного файла (чтобы освободить место на файловой системе).

Можно найти нужный текстовый файл через lsof(посмотреть какой файл держит приложение), восстановить его, после командой

truncate -s 0 filename уменьшить его объем.

4. Занимают ли зомби-процессы какие-то ресурсы в ОС (CPU, RAM, IO)?

зомби процессы блокируют записи в таблицах, который ограничен для каждого пользователя и системы в целом.

зомби процессы приводят к "утечке ресурсов" в виде накопления записей в таблице процессов.

5. В iovisor BCC есть утилита opensnoop:

root@vagrant:~# dpkg -L bpfcc-tools | grep sbin/opensnoop

/usr/sbin/opensnoop-bpfcc

На какие файлы вы увидели вызовы группы open за первую секунду работы утилиты? Воспользуйтесь пакетом bpfcc-tools для Ubuntu 20.04. Дополнительные сведения по установке.

Открыл полотно из :

/usr/share/doc/bpfcc-tools/examples/doc/argdist\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/bashreadline\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/biolatency\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/biosnoop\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/biotop\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/bitesize\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/bpflist\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/btrfsdist\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/btrfsslower\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/cachestat\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/cachetop\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/capable\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/cpudist\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/cpuunclaimed\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/criticalstat\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/dbslower\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/dbstat\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/dcsnoop\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/dcstat\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/deadlock\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/drsnoop\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/execsnoop\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/exitsnoop\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/ext4dist\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/ext4slower\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/filelife\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/fileslower\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/filetop\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/funccount\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/funclatency\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/funcslower\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/gethostlatency\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/hardirqs\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/inject\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/killsnoop\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/klockstat\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/lib

/usr/share/doc/bpfcc-tools/examples/doc/lib/ucalls\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/lib/uflow\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/lib/ugc\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/lib/uobjnew\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/lib/ustat\_example.txt

/usr/share/doc/bpfcc-tools/examples/doc/lib/uthreads\_example.txt

6. Какой системный вызов использует uname -a? Приведите цитату из man по этому системному вызову, где описывается альтернативное местоположение в /proc, где можно узнать версию ядра и релиз ОС.

системный вызов uname выдает информацию по системе.

cat /proc/version

7. Чем отличается последовательность команд через ; и через && в bash? Например:

root@netology1:~# test -d /tmp/some\_dir; echo Hi

Hi

root@netology1:~# test -d /tmp/some\_dir && echo Hi

root@netology1:~#

test -d /tmp/some\_dir; echo Hi, echo Hi будет работать в независимости от результата выполнения первой команды.

test -d /tmp/some\_dir && echo Hi, echo Hi будет только если test -d /tmp/some\_dir завершится с успешным кодом.

8. Есть ли смысл использовать в bash &&, если применить set -e?

Из каких опций состоит режим bash set -euxo pipefail и почему его хорошо было бы использовать в сценариях?

В сценариях когда нужно проверять успешное завершение первой команды, если первая команда завершится со значением больше 0, то сценарий прекратиться и наоборот)

set -o pipefail && command1 | command2 | command3

9. Используя -o stat для ps, определите, какой наиболее часто встречающийся статус у процессов в системе. В man ps ознакомьтесь (/PROCESS STATE CODES) что значат дополнительные к основной заглавной буквы статуса процессов. Его можно не учитывать при расчете (считать S, Ss или Ssl равнозначными).

Больше всего:

I Idle kernel thread