open("/usr/lib/locale/locale-archive", O\_RDONLY|O\_CLOEXEC) = 3

close(3) = 0

open("/etc/magic", O\_RDONLY) = 3

close(3) = 0

open("/usr/share/misc/magic.mgc", O\_RDONLY) = 3

close(3) = 0

open("/usr/lib64/gconv/gconv-modules.cache", O\_RDONLY) = 3

close(3) = 0

write(1, "/dev/tty: character special\n", 28) = 28

munmap(0x7f044708a000, 2285896) = 0

exit\_group(0) = ?

+++ exited with 0 +++

*2. Так где же находится* *база данных утилиты file?*

*open("/usr/lib64/tls/x86\_64/libmagic.so.1", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)*

*stat("/usr/lib64/tls/x86\_64", 0x7ffe4e68b090) = -1 ENOENT (No such file or directory)*

*open("/usr/lib64/tls/libmagic.so.1", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)*

*stat("/usr/lib64/tls", {st\_mode=S\_IFDIR|0555, st\_size=6, ...}) = 0*

*open("/usr/lib64/x86\_64/libmagic.so.1", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)*

*stat("/usr/lib64/x86\_64", 0x7ffe4e68b090) = -1 ENOENT (No such file or directory)*

*open("/usr/lib64/libmagic.so.1", O\_RDONLY|O\_CLOEXEC) = 3*

*read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\340C\0\0\0\0\0\0"..., 832) = 832*

*fstat(3, {st\_mode=S\_IFREG|0755, st\_size=120056, ...}) = 0*

*mmap(NULL, 2213776, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f44dc2a0000*

*mprotect(0x7f44dc2bb000, 2097152, PROT\_NONE) = 0*

*mmap(0x7f44dc4bb000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b000) = 0x7f44dc4bb000*

*close(3) = 0*

*open("/usr/lib64/tls/libz.so.1", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)*

*open("/usr/lib64/libz.so.1", O\_RDONLY|O\_CLOEXEC) = 3*

*read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P!\0\0\0\0\0\0"..., 832) = 832*

*fstat(3, {st\_mode=S\_IFREG|0755, st\_size=90176, ...}) = 0*

*mmap(NULL, 2183272, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f44dc08a000*

*mprotect(0x7f44dc09f000, 2093056, PROT\_NONE) = 0*

*mmap(0x7f44dc29e000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x14000) = 0x7f44dc29e000*

*close(3) = 0*

*open("/usr/lib64/tls/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)*

*open("/usr/lib64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3*

*read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`&\2\0\0\0\0\0"..., 832) = 832*

*mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f44dc6dc000*

*fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2156344, ...}) = 0*

*mmap(NULL, 3985920, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f44dbcbc000*

*mprotect(0x7f44dbe80000, 2093056, PROT\_NONE) = 0*

*mmap(0x7f44dc07f000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c3000) = 0x7f44dc07f000*

*mmap(0x7f44dc085000, 16896, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f44dc085000*

*Может она открывает вот эти библиотеки в поисках данных?*

*3. Еще раз изучите тему перенаправления ввода вывода и файловых дескрипторов и попробуйте ответить на вопрос*

*:>/proc/966/fd/3*

*Cat /dev/null > /proc/966/fd/3*

*Truncate -s 0 /proc/966/fd/3*

*Но только при условии, что, к примеру, процесс open запущен с флагом o\_append, что позволит обнулить размер файла на диске*

*6. Попробуйте прочитать еще раз вопрос и ответить о том какой системный вызов вызывается и где еще можно найти выводимую информацию.*

Речь наверное идёт про библиотечный вызов utsname и если глянуть man 2 uname где-то там есть

Part of the utsname information is also accessible via

*/proc/sys/kernel/*{*ostype*, *hostname*, *osrelease*, *version*,

*domainname*}.

*7. Так имеет смысл использовать или нет? В чем разница при выполнении команд с этими операторами?*

Set -e Простая команда с **ненулевым** статусом выхода вызывает завершение работы оболочки, если только эта простая команда не содержится в **&&** или **||**

*Результат этой команды при использовании && не будет никаким образом виден, то есть оболочка баша не закроется и лично я смысла в этом не вижу.*

*И еще попробуйте нумеровать ответы при выполнении дз, а решение размещать в git репозитории например в файле readme.md*

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.

chdir("/tmp") = 0

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

vagrant@netology1:~$ file /dev/tty

/dev/tty: character special (5/0)

vagrant@netology1:~$ file /dev/sda

/dev/sda: block special (8/0)

vagrant@netology1:~$ file /bin/bash

/bin/bash: ELF 64-bit LSB shared object, x86-64

Используя strace выясните, где находится база данных file на основании которой она делает свои догадки.

Кажется, вот тут вся соль

open("/usr/lib64/gconv/gconv-modules.cache", O\_RDONLY) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=26970, ...}) = 0

mmap(NULL, 26970, PROT\_READ, MAP\_SHARED, 3, 0) = 0x7fc88f07d000

close(3)

1. 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).

Возможно, необходимо отправить stdout процесса в /dev/null

Cat /proc/\*\*\*/fd/3 > /dev/null

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

Википедия: Процесс при завершении (как нормальном, так и в результате не обрабатываемого сигнала) освобождает все свои ресурсы и становится «зомби» — пустой записью в таблице процессов, хранящей статус завершения предназначенный для чтения родительским процессом.

1. 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](https://github.com/iovisor/bcc/blob/master/INSTALL.md).

vagrant@vagrant:~$ sudo opensnoop-bpfcc

PID COMM FD ERR PATH

794 vminfo 6 0 /var/run/utmp

584 dbus-daemon -1 2 /usr/local/share/dbus-1/system-services

584 dbus-daemon 18 0 /usr/share/dbus-1/system-services

584 dbus-daemon -1 2 /lib/dbus-1/system-services

584 dbus-daemon 18 0 /var/lib/snapd/dbus-1/system-services/

1. 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.

The files in the /proc/[pid]/fd directory show the open file descriptors of the process with the PID pid. The files in the /proc/[pid]/fdinfo directory show even more information about these file descriptors. See proc(5) for further details of both of these directories.

1. 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:~#

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

; используется для ввода команд в одной строке, то есть сначала отработает команда 1, после завершения работы отработает команда 2

В случае использования && между командами, вторая будет выполнена только если первая вернет 0.

Set -e Простая команда с **ненулевым** статусом выхода вызывает завершение работы оболочки, если только эта простая команда не содержится в **&&** или **||**

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

Использование set -euxo pipefail  в сценариях позволяет отследить ошибки, которые могут привести в краху скрипта, но не дающие на выходе работы скрипта ненулевых выходных данных.

- o pipefail - Конвейер не завершается до тех пор, пока все компоненты конвейера не будут завершены, а статус выхода конвейера - это значение последней команды для выхода с **ненулевым** статусом выхода или равно **нулю,** если все команды возвращают нулевой статус выхода.

-e Простая команда с **ненулевым** статусом выхода вызывает завершение работы оболочки, если только эта простая команда не содержится в **&&** или **||**

-x Отслеживание исполнения. Оболочка отображает каждую команду после всех раскрытий и перед выполнением, которому предшествует развернутое значение параметра

-u Если этот параметр включен, оболочка отображает сообщение об ошибке, когда пытается раскрыть переменную, которая не установлена.

1. 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).

ps -Ao stat | sort -r | grep -n .

ps -Ao stat | sort | grep -n .

С помощью данных команд видим, что процессов I -43, S – 49

< high-priority (not nice to other users)

N low-priority (nice to other users)

L has pages locked into memory (for real-time and custom IO)

s is a session leader

l is multi-threaded (using CLONE\_THREAD, like NPTL pthreads do)

+ is in the foreground process group.