### **Задание 1**

При каких событиях выполнение процесса переходит в режим ядра?

*Приведите ответ в свободной форме.*

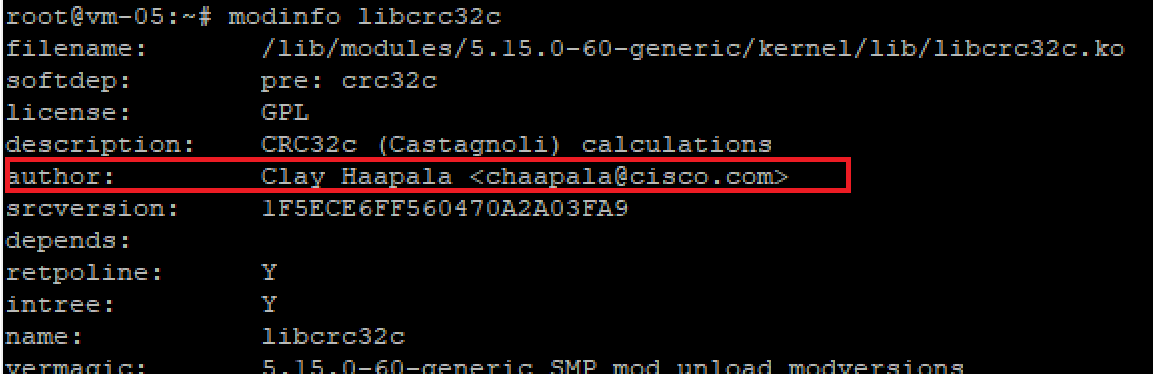
**Ответ:**

1. Когда выполняются системные вызовы от прикладных программ. Но. суть пока не очень понятно. Мы в самом процессе вызываем системные вызовы, но, далее их работа - это код ядра, поэтому не очень понятно причем тут код процесса. Я бы так сказал - что видимо это продолжение выполнение процесса по смыслу, но, работает код ядра - типа функция, с параметрами из процесса пользователя.
2. Когда срабатывает прерывание аппаратное (или возможно - от таймера видимо), то, вызывается обработчик прерываний, и там видимо тоже может быть код от процесса? Тут не очень понятно при чем тут процесс пользователя. Могу только предположить, что срабатывает например таймер, и по таймеру на ядро нужно поместить другой процесс, но, это все же будет работа самого процесса - его кода, не в режиме ядра. Как при прерывании может выполняться процесс пользователя - не понятно пока.

### **Задание 2**

Найдите имя автора модуля libcrc32c.

*В качестве ответа приложите снимок экрана с выводом команды.*



### **Задание 3**

Используя утилиту strace, выясните, какой системный вызов использует команда cd для того, чтобы сменить директорию.

Примечание:

1. Команда cd не является внешним файлом, но для наших целей можно использовать: strace bash -c 'cd /tmp'.
2. При выводе strace вы можете увидеть много системных вызовов, чтобы разобраться, за что отвечает каждый из них, можете воспользоваться встроенной помощью man.

*В качестве ответа напишите название системного вызова.*

Я нашел два системных вызова:

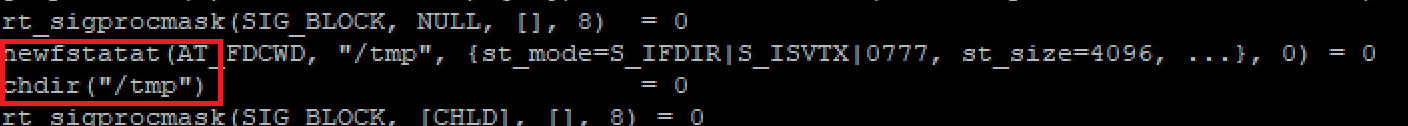
newfstatat(AT\_FDCWD, "/tmp", {st\_mode=S\_IFDIR|S\_ISVTX|0777, st\_size=4096, ...}, 0) = 0

chdir("/tmp")

Поэтому названия:

1. newfstatat
2. chdir

Цитирую:



Было замечание цитирую:

Получится сократить варианты до одного? :) И системных вызовов в данном случае будет гораздо больше чем два.

**Дополняю:**

Всего системных вызовов конечно больше, но, я искал только те в которых есть путь “/tmp” то их только два и они рядом (и их и привел). Поэтому, считаю, логично считать что оба вызова рождемы командой “cd”.

“chdir” - видна суть из названия вызова и из документации - изменить рабочий каталог. Это единственный ответ (если нужно один).

Но, работает реально, он в связке с предыдущим “newfstatat” - считывает состояние файла. А потом уже меняет директорию другим системным вызовом chdir.

## **Дополнительные задания (со звездочкой\*)**

Эти задания дополнительные (необязательные к выполнению) и никак не повлияют на получение вами зачета по этому домашнему заданию. Вы можете их выполнить, если хотите глубже и/или шире разобраться в материале.

### **Задание 4\***

**Соберите свой модуль и загрузите его в ядро.**

Примечание: лучше использовать чистую виртуальную машину, чтобы нивелировать шанс сломать систему.

**1) Установим необходимые пакеты:**

apt-get install gcc make linux-headers-$(uname -r)

**2) Создаем файл модуля:**

mkdir kmod-hello\_world

cd kmod-hello\_world/

touch ./mhello.c

#define MODULE

#include <linux/module.h>

#include <linux/init.h>

#include <linux/kernel.h>

MODULE\_LICENSE("GPLv3");

int init\_module(void){

printk("<1> Hello,World\n");

return 0;

}

void cleanup\_module(void){

printk("<1> Goodbye.\n");

}

**3) Создаем Makefile:** touch ./Makefile

obj-m += mhello.o

hello-objs := mhello.c

all:

make -C /lib/modules/$(shell uname -r)/build/ M=$(PWD) modules

clean:

make -C /lib/modules/$(shell uname -r)/build/ M=$(PWD) clean

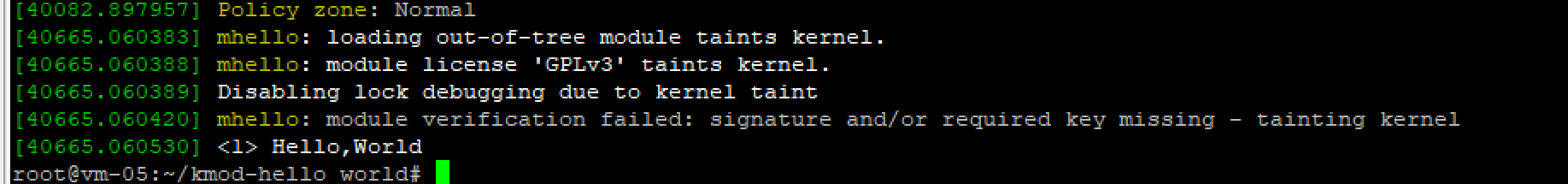
*Обратите внимание, что отступы перед make - это табуляция, а не пробелы. Для синтаксиса Makefile это важно.*

**4) Собираем модуль и устанавливаем его с помощью insmod.**

make all

insmod path/to/module.ko

*В качестве ответа приложите скриншот вывода установки модуля в dmesg.*

**