Examen 1

Universidad ICESI
Curso: Sistemas Operativos
Docente: Daniel Barragán C.
Tema: Comandos de Linux, Virtualización

Nombre: Alejandro Bueno Cardona.

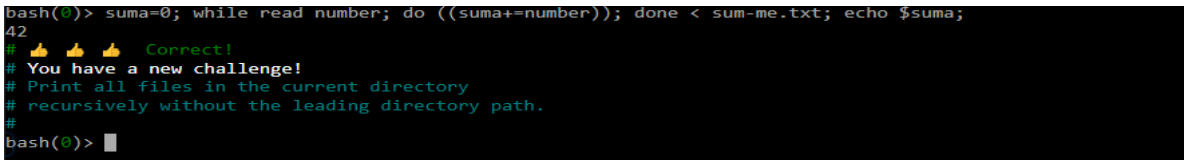
Código: A00335472

URL: https://github.com/abc1196/so-exam1

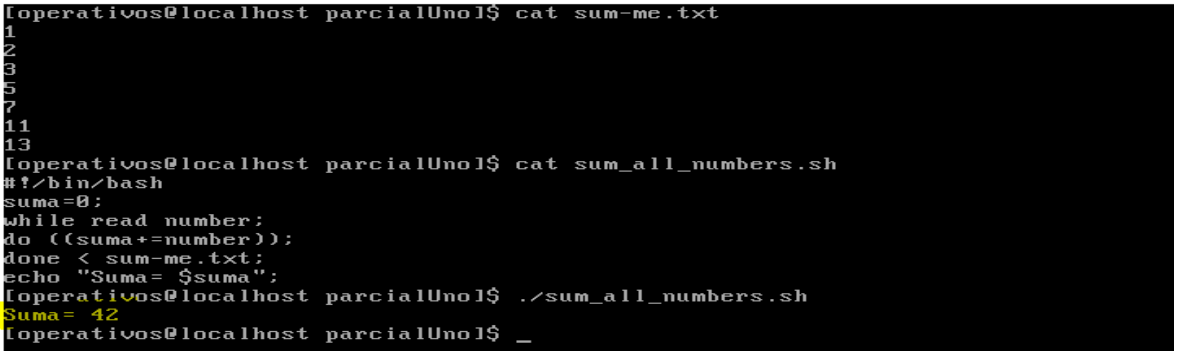
Retos CMD Challenge

1. **Sum_all_numbers**

   Para el primer reto se creó un script para ejecutar la tarea. El script inicia declarando la variable **suma** igual a cero. Luego, un loop (**while**) se encarga de leer cada línea del archivo. Este valor es agregado a la variable suma, que es mostrada, al final del bucle, mediante el comando **echo.**
   CMD



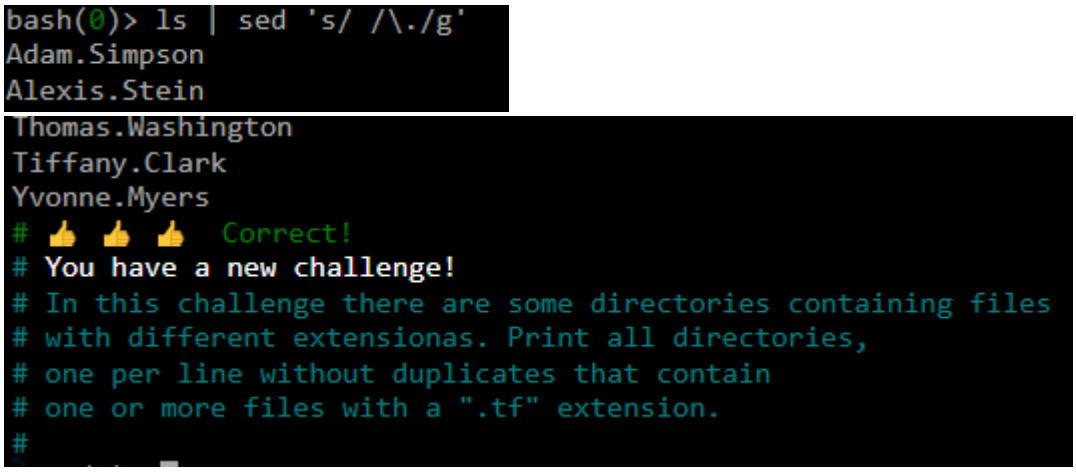
   CentOS7



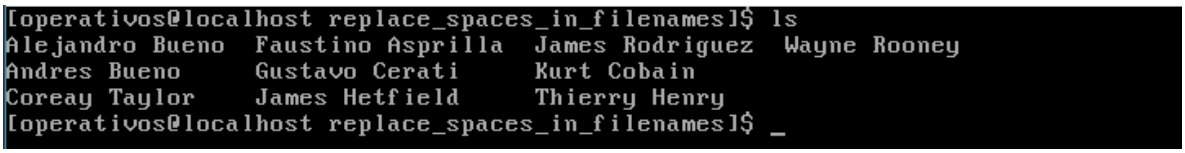
2. Replace_spaces_in_filenames

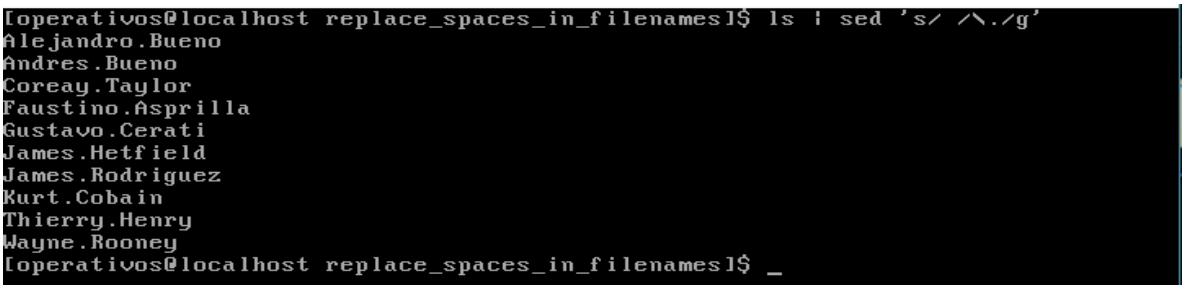
En el segundo reto se implementó uno de los comandos vistos en clase: **sed**. En este caso, se tomaron los archivos del directorio actual con **ls**. Posteriormente, se utilizó el comando **sed** reemplazar los espacios (“/ /”) con puntos (“\.”).

CMD



CentOS7

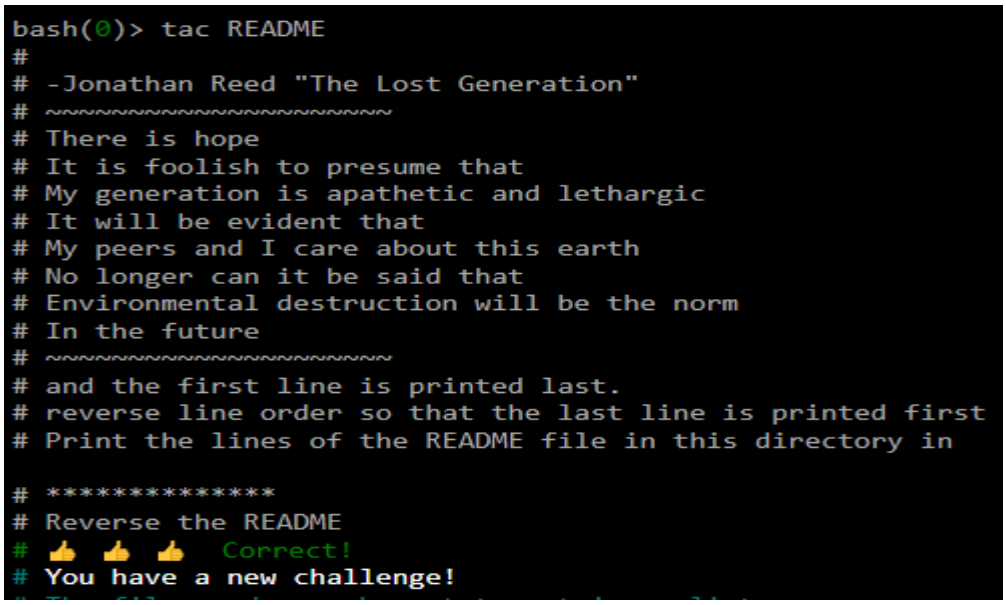




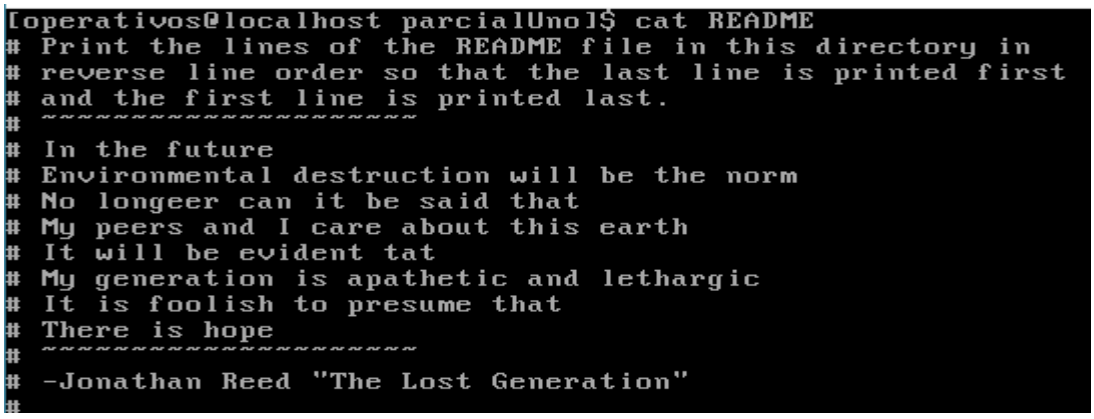
3. Reverse_readme

El tercer reto no tuvo mayor dificultad. El comando **cat**, utilizado para ver el contenido de los archivos, tiene su inverso: **tac**, que permite mostrar el contenido de un archivo en sentido contrario, es decir, la primera línea se imprime de última y la última línea de primera.

CMD



CentOS7



```
[operativos@localhost parcialUno]$ tac README
#
# -Jonathan Reed "The Lost Generation"
# ~~~~~~~~~~~~~~~~~~~~~~
# There is hope
# It is foolish to presume that
# My generation is apathetic and lethargic
# It will be evident tat
# My peers and I care about this earth
# No longeer can it be said that
# Environmental destruction will be the norm
# In the future
# ~~~~~~~~~~~~~~~~~~~~~~
# and the first line is printed last.
# reverse line order so that the last line is printed first
# Print the lines of the README file in this directory in
[operativos@localhost parcialUno]$ _
```

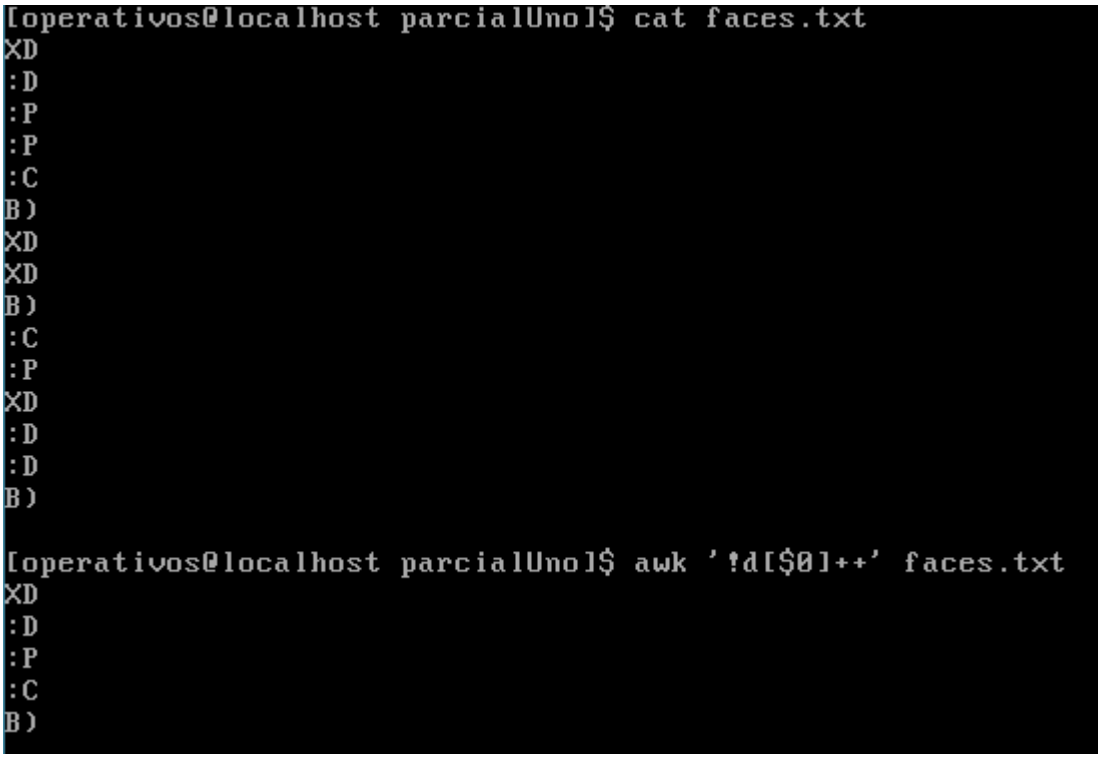
4. **Remove_duplicated_lines**

   En el cuarto reto se buscó e implementó el comando **awk**. El parámetro **'!duplicate[$0]++'** le indica al comando que líneas imprimir. **$0** contiene toda la línea. Si el contenido del índice del arreglo, en este caso **duplicate**, no ha sido establecido, entonces el índice **duplicate** se incremente y la línea se imprime.

   CMD

```
bash(0)> awk '!duplicate[$0]++' faces.txt
(◕‿◕)
(^^)
0_0
ಠ_ಠ
¯\(°_o)/¯
( ゜ワ゜)
(╯‵□′)╯
# 👍 👍 👍  Correct!
# You have a new challenge!
```

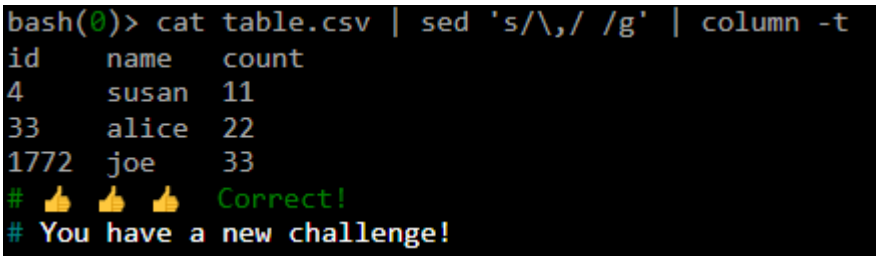
CentOS7



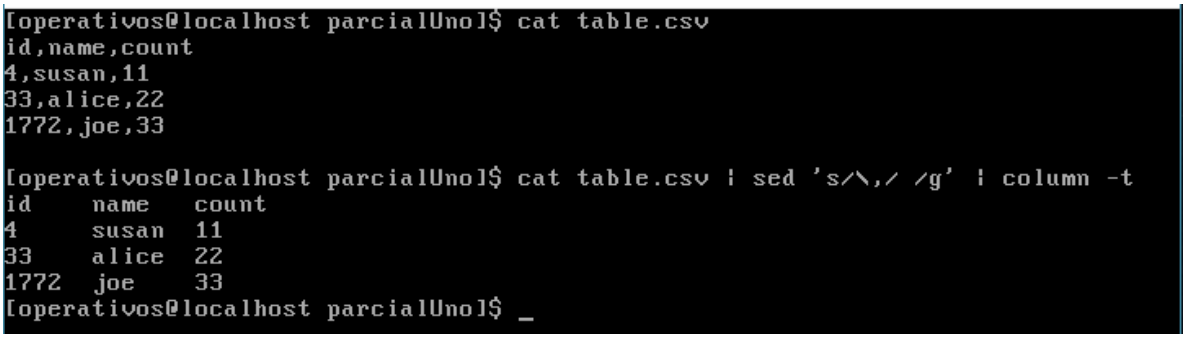
5. **Disp_table**

En el quinto reto se implementó uno de los comandos vistos en clase de nuevo: **sed**. En este caso, se muestra el contenido del archivo table.csv. Posteriormente, se utilizó el comando **sed** reemplazar las comas (“\,”) con espacios (“/ /”). Por último, el comando **column** permite ordenar la información del archivo en formato de columnas. El parámetro **-t** tabula simple a cada una.

CMD



CentOS7


[operativos@localhost parcialUno]$ cat table.csv
id,name,count
4,susan,11
33,alice,22
1772,joe,33
[operativos@localhost parcialUno]$ cat table.csv | sed 's/\,/ /g' | column -t
id name count
4 susan 11
33 alice 22
1772 joe 33
[operativos@localhost parcialUno]$ _

Script Gutenberg-Crontab

En el momento que se realizó el parcial, Gutenberg contaba con 66588 libros en su página. Cada uno se obtiene mediante el enlace https://www,gutenberg.org/files/numID/numID.txt. Sin embargo, a partir del numID=40, los libros presentan variaciones en su identificador. Por esto, el script presenta, de manera aleatoria, los primeros 40 libros, es decir, el conjunto de libros con numID entre 1 y 40. El script que realiza la descarga del libro en el directorio **/home/Gutenberg/mybooks** es el siguiente:

```
[root@localhost scripts]# ls
downloadGutenbergBook.sh
[root@localhost scripts]# cat downloadGutenbergBook.sh
#!/bin/bash
numID=$((RANDOM%39));
hora=$(date +%H:%M)
URL="https://www.gutenberg.org/files/$numID/$numID.txt"
rm -f /home/gutenberg/mybooks/*
wget -O  /home/gutenberg/mybooks/fiveMinuteBook$numID-$hora.txt $URL
[root@localhost scripts]# _
```

El script asigna dos variables: numID, con el identificador aleatorio mencionado anteriormente; hora, con la hora actual en formato HH:MM. Luego, se crea el URL, se elimina el libro anterior (comando **rm)** y, finalmente, se descarga el libro con el comando **wget**, cuyo parámetro **-O** almacena la descarga en el directorio **/home/Gutenberg/mybooks** y en el archivo **fiveMinuteBook$numID-$hora.txt**.

Por otro lado, para ejecutar el script cada 5 minutos, se utilizó **crontab**, que es un archivo de texto, donde se guarda una lista de comandos para ser ejecutados. A continuación, se puede ver el formato para los comandos de **crontab**:

```
* * * * * comando ha ser ejecutado
- - - - -
| | | | |
| | | | ----- Día de la semana (0 - 7)
| | | ------- Mes (1 - 12)
| | --------- Día del mes (1 - 31)
| ----------- Hora (0 - 23)
------------- Minuto (0 - 59)
```

Tomada de http://www.desarrollolibre.net/blog/tema/106/linux/ejecutar-script-automaticamente-con-cron-en-linux#.WdhTJ2jWxEY

Cambiando el primero parámetro de minutos a ***/5** y dejando en ***** los demás, **crontab** procede a ejecutar el script cada 5 minutos.

```
[root@localhost ~]# crontab -l
*/5 * * * * /root/scripts/downloadGutenbergBook.sh
```

En el momento en que se guardó el archivo **crontab**, llegó notificación al usuario **root** con la siguiente información:

```
X-Cron-Env: <LANG=es_CO.UTF-8>
X-Cron-Env: <SHELL=/bin/sh>
X-Cron-Env: <HOME=/root>
X-Cron-Env: <PATH=/usr/bin:/bin>
X-Cron-Env: <LOGNAME=root>
X-Cron-Env: <USER=root>
Message-Id: <20171007023002.90F7FC7BFA7@localhost.localdomain>
Date: Fri,  6 Oct 2017 21:30:01 -0500 (COT)

--2017-10-06 21:30:01--  https://www.gutenberg.org/files/3/3.txt
Resolviendo www.gutenberg.org (www.gutenberg.org)... 152.19.134.47, 2610:28:3090
:3000:0:bad:cafe:47
Conectando con www.gutenberg.org (www.gutenberg.org)[152.19.134.47]:443... conec
tado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 16384 (16K) [text/plain]
Grabando a: “/home/gutenberg/mybooks/fiveMinuteBook3-21:30.txt"

     0K .......... ......                                    100%  119K=0,1s

2017-10-06 21:30:02 (119 KB/s) - “/home/gutenberg/mybooks/fiveMinuteBook3-21:30.
txt" guardado [16384/16384]
```

En la anterior imagen, se evidenció que el script se ejecuta y almacena el nuevo archivo: fiveMinuteBook-21:30.txt. Este archivo tiene numID=3 y fue descargado a las 21:30 (hora de la VM).

Pasados 5 minutos, llegó notificación al usuario **root** con la nueva descarga:

```
[root@localhost ~]# date
vie oct  6 21:34:16 COT 2017
[root@localhost ~]#
Tiene correo nuevo en /var/spool/mail/root
[root@localhost ~]# _
```

```
--2017-10-06 21:35:01--  https://www.gutenberg.org/files/20/20.txt
Resolviendo www.gutenberg.org (www.gutenberg.org)... 152.19.134.47, 2610:28:3090
:3000:0:bad:cafe:47
Conectando con www.gutenberg.org (www.gutenberg.org)[152.19.134.47]:443... conec
tado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 507133 (495K) [text/plain]
Grabando a: “/home/gutenberg/mybooks/fiveMinuteBook20-21:35.txt"

     0K .......... .......... .......... .......... .......... 10%  173K 3s
    50K .......... .......... .......... .......... .......... 20%  193K 2s
   100K .......... .......... .......... .......... .......... 30%  249K 2s
   150K .......... .......... .......... .......... .......... 40%  320K 1s
   200K .......... .......... .......... .......... .......... 50%  312K 1s
   250K .......... .......... .......... .......... .......... 60%  285K 1s
   300K .......... .......... .......... .......... .......... 70%  314K 1s
   350K .......... .......... .......... .......... .......... 80%  303K 0s
   400K .......... .......... .......... .......... .......... 90% 84,5K 0s
   450K .......... .......... .......... .......... .....     100%  154K=2,5s

2017-10-06 21:35:05 (202 KB/s) - “/home/gutenberg/mybooks/fiveMinuteBook20-21:35
.txt" guardado [507133/507133]
```

Ahora, se guardó el archivo con numID=20 a las 21:35 (hora de la VM).

Para terminar, se comprobó la existencia del archivo en el usuario y directorio requerido:

```
[gutenberg@localhost mybooks]$ ls
fiveMinuteBook20-21:35.txt
[gutenberg@localhost mybooks]$ pwd
/home/gutenberg/mybooks
[gutenberg@localhost mybooks]$ _
```

Rickroll.c

En términos generales, el código fuente rickroll.c carga un módulo del kernel, que cambia la ejecución de los archivos en formato .mp3. Cuando el usuario lo abre, se reproduce, en cambio, la canción *"Never Gonna Give You Up"* de Rick Astley. A continuación, se explica, de manera general, el código correspondiente.

Primero, se incluyen las librerías referentes a Linux: module, kernel, init, syscalls y string. Además, se establece la Licencia Pública General (GPL) y su autor.

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/syscalls.h>
#include <linux/string.h>


MODULE_LICENSE("GPL");
MODULE_AUTHOR("Kamal Marhubi");
MODULE_DESCRIPTION("Rickroll module");
```

Segundo, se establece la ruta de la canción con **rickroll_filename**. Luego, se establece el parámetro del módulo para la ruta del archivo. En tiempo de ejecución, **insmod** enviara las variables al módulo del kernel. Los argumentos son: el nombre de la variable, el tipo de dato y los permisos correspondiente para el archivo en sysfs, que permite configurar parámetros al kernel.

```
static char *rickroll_filename = "/home/bork/media/music/Rick Astley - Never Gonna Give You Up.mp3";

/*
 * Set up a module parameter for the filename. The arguments are variable name,
 * type, and permissions The third argument is the permissions for the parameter
 * file in sysfs, something like
 *
 * /sys/module/<module_name>/parameters/<parameter_name>
 *
 * We're setting it writeable by root so it can be modified without reloading
 * the module.
 */
module_param(rickroll_filename, charp, S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH);
MODULE_PARM_DESC(rickroll_filename, "The location of the rick roll file");

```

Tercero, se crean dos constantes para deshabilitar y habilitar el área de memoria, que incluye la **system call table**. Porque modificar esta área puede ocasionar errores de protección. Después, se crean las funciones encargadas de encontrar la **system call table**, abrir la nueva canción y ejecutar el archivo original. Las tres tienen la particularidad de ser declaradas como *asmlinkage*. Esta etiqueta le indica al compilador, que busque en la pila de la CPU los parámetros de las funciones, en vez de hacerlo en los registros. Estas funciones, tipo *system calls*, guardan sus parámetros en la pila, por lo que es necesario informarle al compilador al respecto con *asmlinkage*.

```
4   #define DISABLE_WRITE_PROTECTION (write_cr0(read_cr0() & (~ 0x10000)))
5   #define ENABLE_WRITE_PROTECTION (write_cr0(read_cr0() | 0x10000))
6
7
8   static unsigned long **find_sys_call_table(void);
9   asmlinkage long rickroll_open(const char __user *filename, int flags, umode_t mode);
0
1   asmlinkage long (*original_sys_open)(const char __user *, int, umode_t);
2   asmlinkage unsigned long **sys_call_table;
```

Cuarto, carga el módulo al sistema con **module_init**, cuyo parámetro es la función **rickroll_init**, que primero valida la existencia de la canción y la **system call table**. La tabla es cargada llamando la función **find_sys_call_table**, que, de manera general, busca en el espacio de memoria del kernel, su dirección. Seguido, la función reemplaza la entrada para ser abierto con la función **rickroll_open**. La ubicación de la llamada al sistema original se guarda, para ser restablecida después.

```
static unsigned long **find_sys_call_table() {
    unsigned long offset;
    unsigned long **sct;

    for(offset = PAGE_OFFSET; offset < ULLONG_MAX; offset += sizeof(void *)) {
        sct = (unsigned long **) offset;

        if(sct[__NR_close] == (unsigned long *) sys_close)
            return sct;
    }

    /*
     * Given the loop limit, it's somewhat unlikely we'll get here. I don't
     * even know if we can attempt to fetch such high addresses from memory,
     * and even if you can, it will take a while!
     */
    return NULL;
}
```

```
module_init(rickroll_init);
```

```
static int __init rickroll_init(void)
{
    if(!rickroll_filename) {
        printk(KERN_ERR "No rick roll filename given.");
        return -EINVAL;  /* invalid argument */
    }

    sys_call_table = find_sys_call_table();

    if(!sys_call_table) {
        printk(KERN_ERR "Couldn't find sys_call_table.\n");
        return -EPERM;  /* operation not permitted; couldn't find general error */
    }

    /*
     * Replace the entry for open with our own function. We save the location
     * of the real sys_open so we can put it back when we're unloaded.
     */
    DISABLE_WRITE_PROTECTION;
    original_sys_open = (void *) sys_call_table[__NR_open];
    sys_call_table[__NR_open] = (unsigned long *) rickroll_open;
    ENABLE_WRITE_PROTECTION;

    printk(KERN_INFO "Never gonna give you up!\n");
    return 0;  /* zero indicates success */
}
```

La función **rickroll_open** primero valida que el archivo sea tipo .mp3. En caso contrario, retorna la llamada al sistema original. Si es válido, entonces la función procede a ejecutar el archivo rickroll, de manera específica, en la siguiente línea:

*fd = (*original_sys_open)(rickroll_filename, flags, mode);*

```
asmlinkage long rickroll_open(const char __user *filename, int flags, umode_t mode)
{
    int len = strlen(filename);

    /* See if we should hijack the open */
    if(strcmp(filename + len - 4, ".mp3")) {
        /* Just pass through to the real sys_open if the extension isn't .mp3 */
        return (*original_sys_open)(filename, flags, mode);
    } else {
        /* Otherwise we're going to hijack the open */
        mm_segment_t old_fs;
        long fd;

        /*
         * sys_open checks to see if the filename is a pointer to user space
         * memory. When we're hijacking, the filename we pass will be in kernel
         * memory. To get around this, we juggle some segment registers. I
         * believe fs is the segment used for user space, and we're temporarily
         * changing it to be the segment the kernel uses.
         *
         * An alternative would be to use read_from_user() and copy_to_user()
         * and place the rickroll filename at the location the user code passed
         * in, saving and restoring the memory we overwrite.
         */
        old_fs = get_fs();
        set_fs(KERNEL_DS);

        /* Open the rickroll file instead */
        fd = (*original_sys_open)(rickroll_filename, flags, mode);

        /* Restore fs to its original value */
        set_fs(old_fs);

        return fd;
    }
```

Por último, se cierra el módulo con **module_exit**, que tiene de parámetro la función **rickroll_cleanup**, que reestablece la llamada al sistema original en la **system call table**.

```
module_exit(rickroll_cleanup);
```

```
static void __exit rickroll_cleanup(void)
{
    printk(KERN_INFO "Ok, now we're gonna give you up. Sorry.\n");

    /* Restore the original sys_open in the table */
    DISABLE_WRITE_PROTECTION;
    sys_call_table[__NR_open] = (unsigned long *) original_sys_open;
    ENABLE_WRITE_PROTECTION;
}
```

Referencias

- https://cmdchallenge.com
- https://www.gutenberg.org
- https://github.com/jvns/kernel-module-fun/blob/master/rickroll.c
- https://www.youtube.com/watch?v=efEZZZf_nTc
- http://www.desarrollolibre.net/blog/tema/106/linux/ejecutar-script-automaticamente-con-cron-en-linux#.WdhTJ2jWxEY
- https://www.quora.com/Linux-Kernel-What-does-asmlinkage-mean-in-the-definition-of-system-calls
- https://github.com/ICESI/so-commands/tree/master/centos7
- https://lists.debian.org/debian-user-spanish/2011/08/msg00272.html
- https://www.tutorialspoint.com/unix_commands/awk.htm