Hacettepe University Computer Engineering

BBM 459 Secure Programming Laboratory Programming Assignment 1

Environment Variable and Set-UID Program

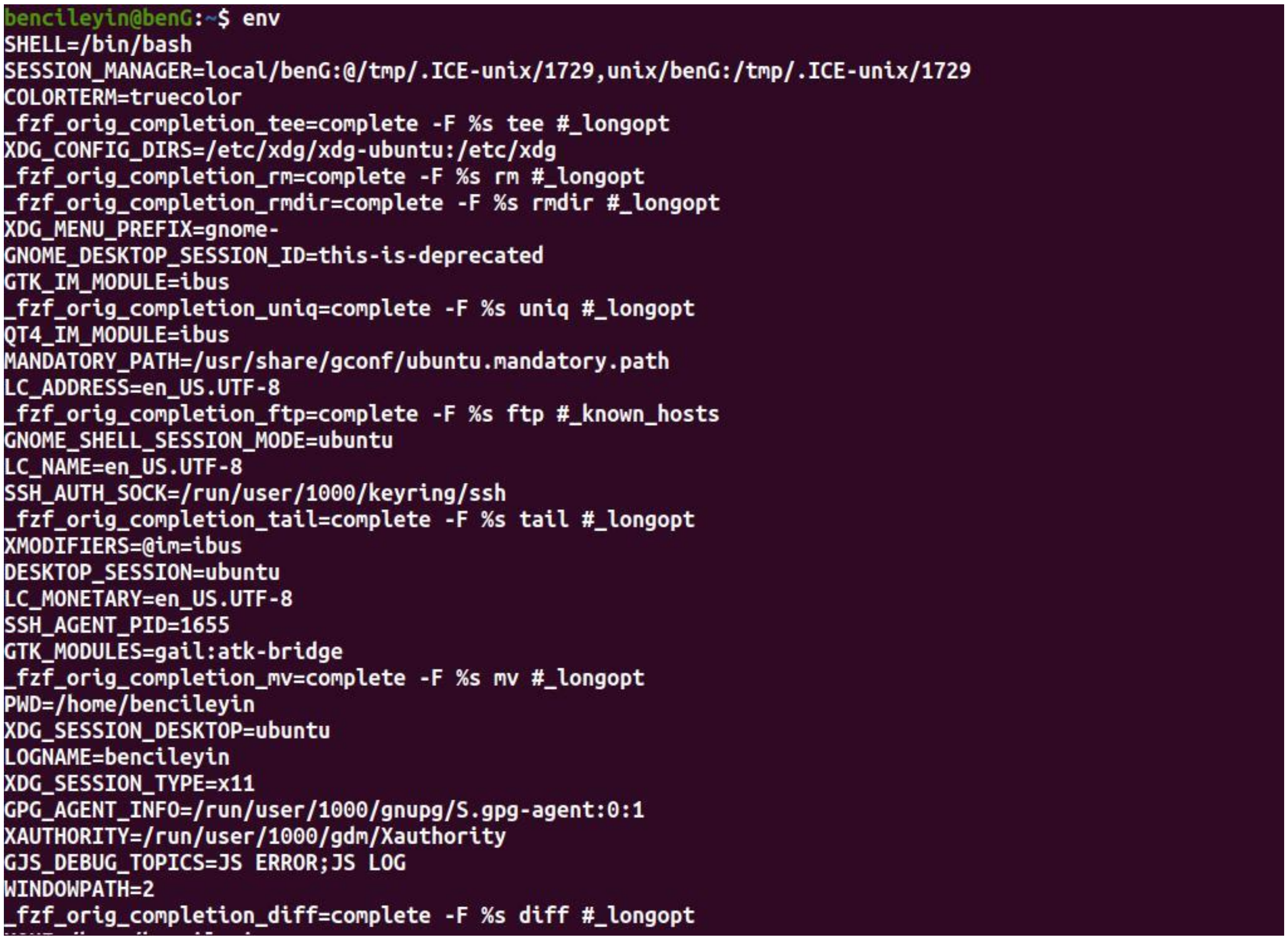
Spring 2021

Ahmet Hakan YILDIZ

Cihan KÜÇÜK

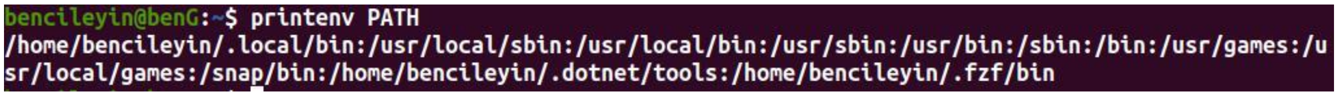
Task 1: Manipulating Environment Variables

env command prints all environment variables.



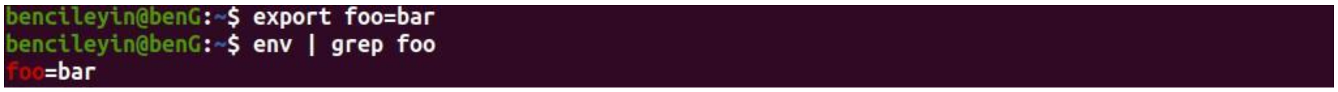
```
bencileyin@benG:~$ env
SHELL=/bin/bash
SESSION_MANAGER=local/benG:@/tmp/.ICE-unix/1729,unix/benG:/tmp/.ICE-unix/1729
COLORTERM=truecolor
_fzf_orig_completion_tee=complete -F %s tee #_longopt
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
_fzf_orig_completion_rm=complete -F %s rm #_longopt
_fzf_orig_completion_rmdir=complete -F %s rmdir #_longopt
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
_fzf_orig_completion_uniq=complete -F %s uniq #_longopt
QT4_IM_MODULE=ibus
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
LC_ADDRESS=en_US.UTF-8
_fzf_orig_completion_ftp=complete -F %s ftp #_known_hosts
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=en_US.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
_fzf_orig_completion_tail=complete -F %s tail #_longopt
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=en_US.UTF-8
SSH_AGENT_PID=1655
GTK_MODULES=gail:atk-bridge
_fzf_orig_completion_mv=complete -F %s mv #_longopt
PWD=/home/bencileyin
XDG_SESSION_DESKTOP=ubuntu
LOGNAME=bencileyin
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
_fzf_orig_completion_diff=complete -F %s diff #_longopt
```

printenv PATH command prints a specific environment variable. ***PATH*** variable is printed as follows:



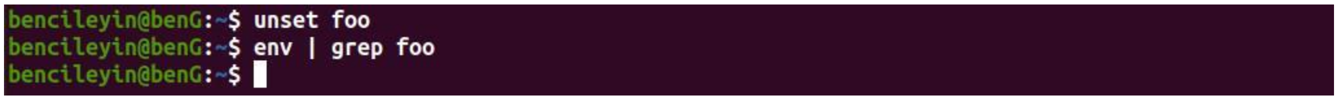
```
bencileyin@benG:~$ printenv PATH
/home/bencileyin/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/u
sr/local/games:/snap/bin:/home/bencileyin/.dotnet/tools:/home/bencileyin/.fzf/bin
```

export command sets an environment variable. A new variable called foo is set to bar.



```
bencileyin@benG:~$ export foo=bar
bencileyin@benG:~$ env | grep foo
foo=bar
```

When unset foo command is executed, foo is removed from environment variables list.

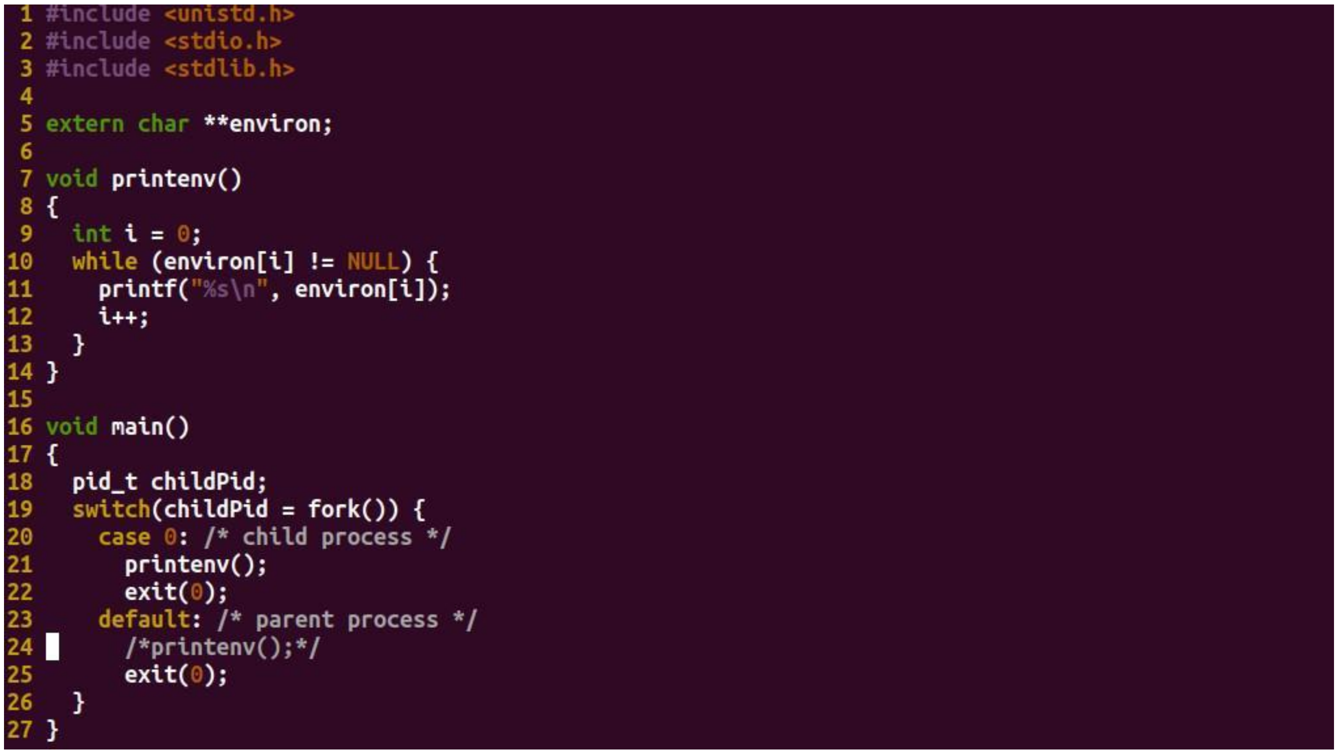


```
bencileyin@benG:~$ unset foo
bencileyin@benG:~$ env | grep foo
bencileyin@benG:~$
```

Task 2: Inheriting environment variables from parents

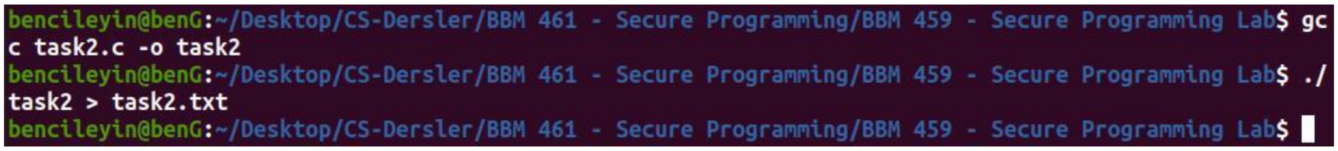
Step 1:

The given source code is created as ***task2.c*** and compiled into ***task2***.



```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

void printenv()
{
  int i = 0;
  while (environ[i] != NULL) {
    printf("%s\n", environ[i]);
    i++;
  }
}

void main()
{
  pid_t childPid;
  switch(childPid = fork()) {
    case 0: /* child process */
      printenv();
      exit(0);
    default: /* parent process */
      /*printenv();*/
      exit(0);
  }
}
```

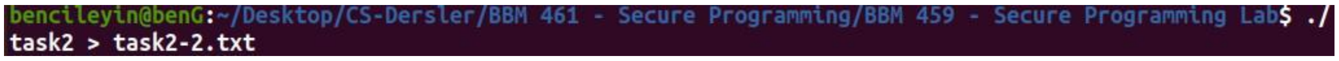
The output of the program is written into ***task2.txt***



```
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$ gc
c task2.c -o task2
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$ ./
task2 > task2.txt
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
```

Step 2:

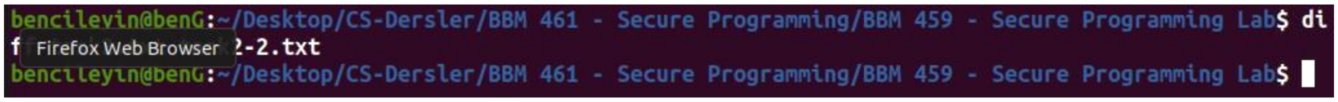
After child process is commented out and parent processes is uncommented, the source code is compiled again and this time its output is written into ***task2-2.txt***.



```
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$ ./
task2 > task2-2.txt
```

Step 3:

Their difference is checked using ***diff*** command. No difference is observed. This means that, child processes have same environment variables as their parent processes.

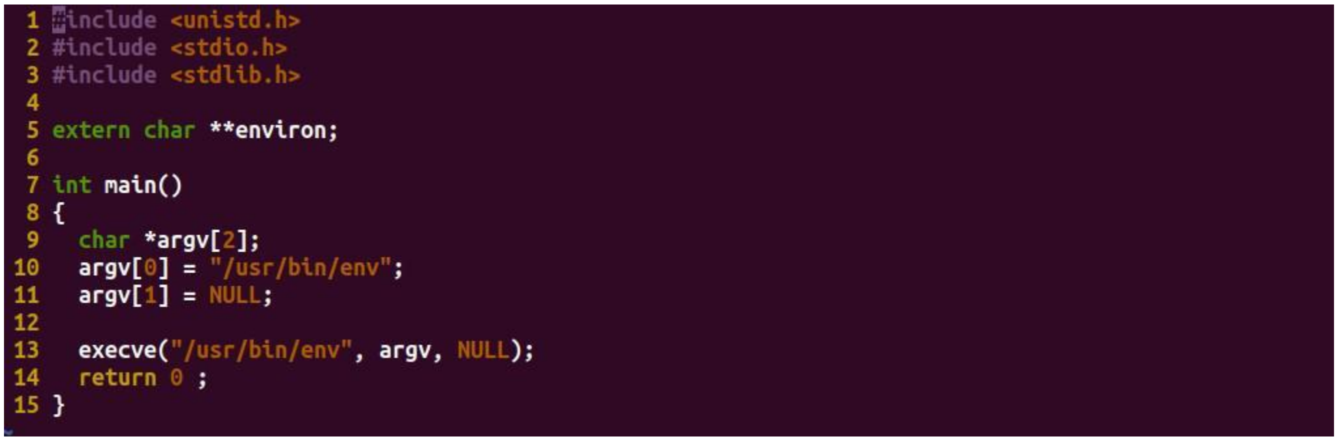


```
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$ di
f Firefox Web Browser 2-2.txt
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$ 
```

Task 3: Environment variables and "execve()"

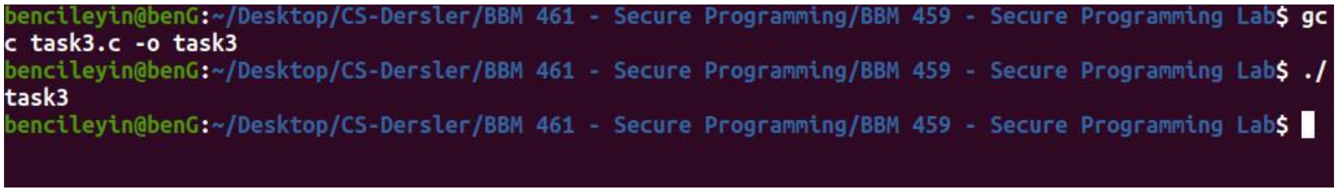
Step 1:

The given source code is created as ***task3.c***



```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

int main()
{
  char *argv[2];
  argv[0] = "/usr/bin/env";
  argv[1] = NULL;

  execve("/usr/bin/env", argv, NULL);
  return 0 ;
}
```

Afterwards, this program is compiled as ***task3*** and executed.

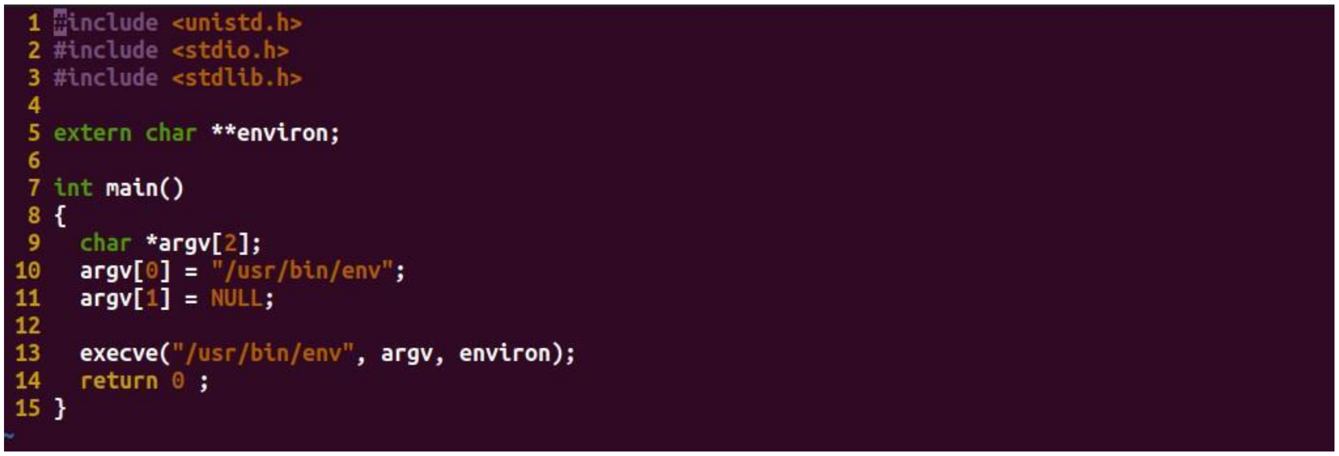


```
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$ gc
c task3.c -o task3
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$ ./
task3
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$ 
```

No output is printed because ***NULL*** is sent as the environment variables list.

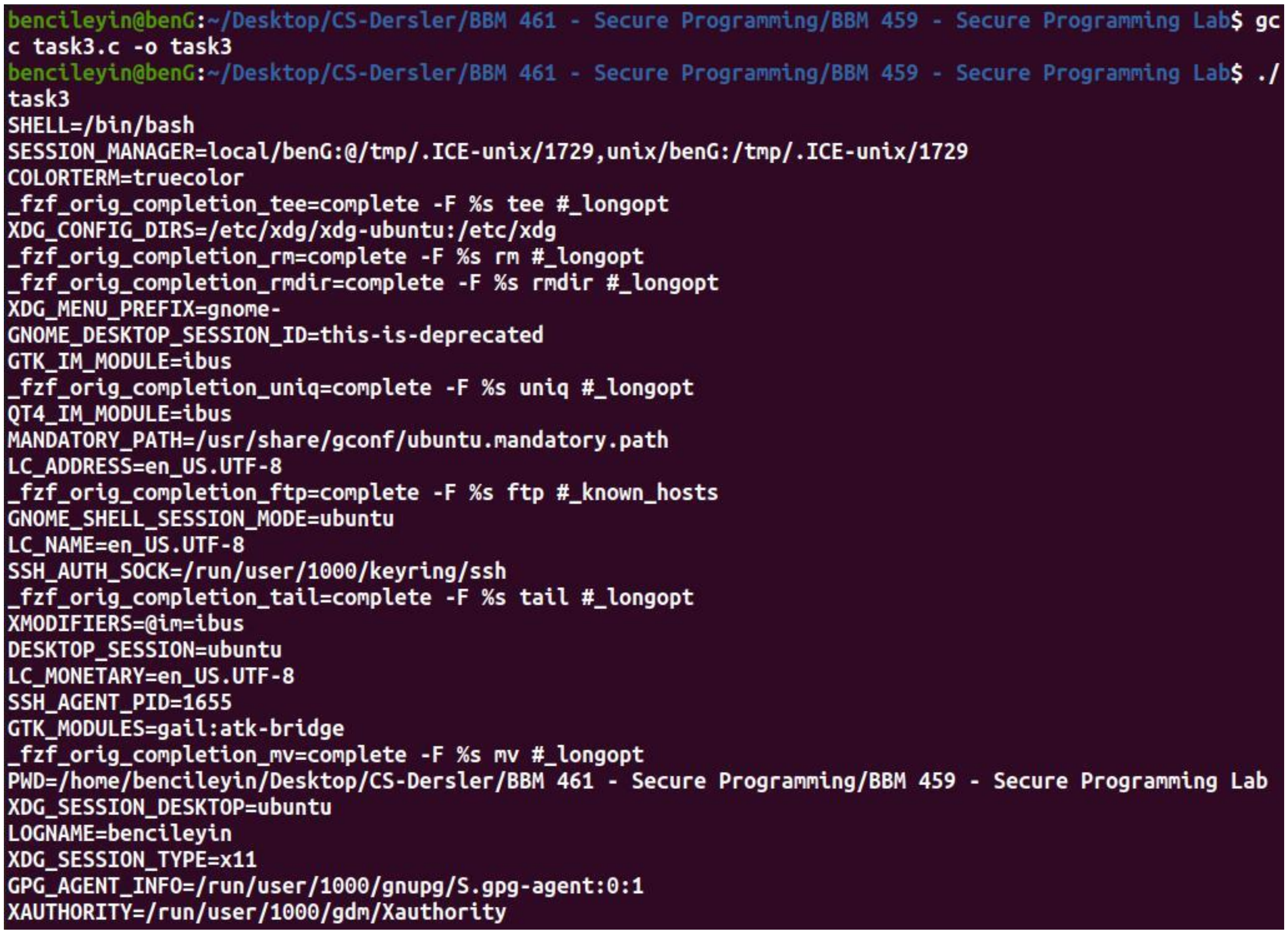
Step 2:

In this step, the ***environ*** variable is sent to ***execve*** function. That makes environment variable of the new process same as ***task3***'s environment variables.



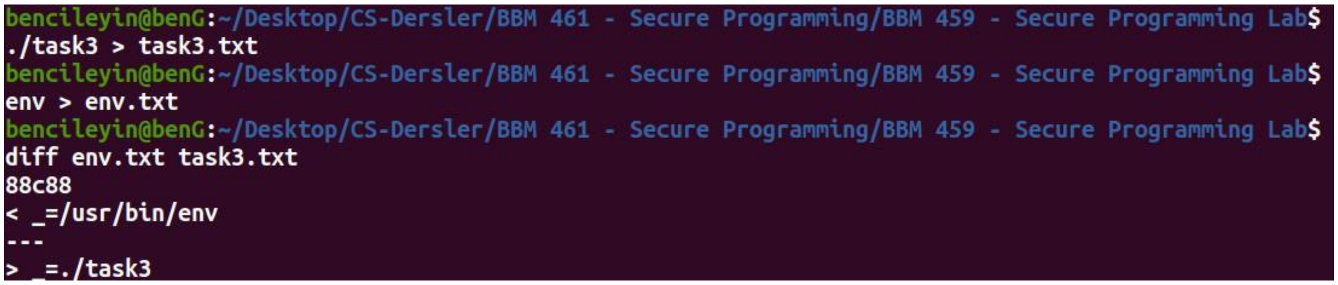
```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

int main()
{
  char *argv[2];
  argv[0] = "/usr/bin/env";
  argv[1] = NULL;

  execve("/usr/bin/env", argv, environ);
  return 0 ;
}
```

Afterwards, this program is compiled as ***task3*** and executed.



```
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$ gc
c task3.c -o task3
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$ ./
task3
SHELL=/bin/bash
SESSION_MANAGER=local/benG:@/tmp/.ICE-unix/1729,unix/benG:/tmp/.ICE-unix/1729
COLORTERM=truecolor
_fzf_orig_completion_tee=complete -F %s tee #_longopt
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
_fzf_orig_completion_rm=complete -F %s rm #_longopt
_fzf_orig_completion_rmdir=complete -F %s rmdir #_longopt
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
_fzf_orig_completion_uniq=complete -F %s uniq #_longopt
QT4_IM_MODULE=ibus
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
LC_ADDRESS=en_US.UTF-8
_fzf_orig_completion_ftp=complete -F %s ftp #_known_hosts
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=en_US.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
_fzf_orig_completion_tail=complete -F %s tail #_longopt
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=en_US.UTF-8
SSH_AGENT_PID=1655
GTK_MODULES=gail:atk-bridge
_fzf_orig_completion_mv=complete -F %s mv #_longopt
PWD=/home/bencileyin/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab
XDG_SESSION_DESKTOP=ubuntu
LOGNAME=bencileyin
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
```

The output is complex, therefore output is written into ***task3.txt*** file and compared with ***env*** command's regular output. ***env*** output is printed to ***env.txt*** and compared with ***task3.txt*** via ***diff*** command.



```
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
./task3 > task3.txt
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
env > env.txt
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
diff env.txt task3.txt
88c88
< _=/usr/bin/env
---
> _=./task3
```

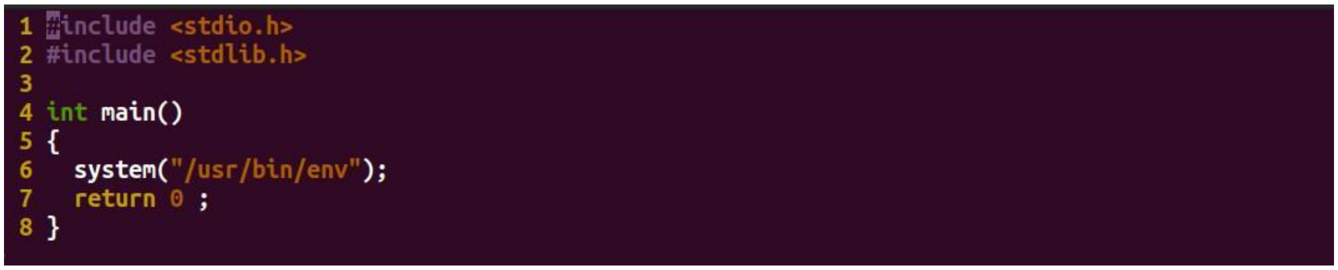
The only difference is the the variable called **_**. After some research, it is found that ***$_*** is not an environment variable but instead a special bash variable showing the argument of the last command. Therefore, we can safely say that there is no difference in environment variables of the parent process and child process.

Step 3:

In conclusion, when ***execve*** function is called, the child process gets its environment variables through the third argument of the function. If ***NULL*** value is sent as third argument, then there are no environment variables in child process. Otherwise, the third argument is a pointer to environment variables list whoso type is ***char*****. It is a pointer to pointer to characters.

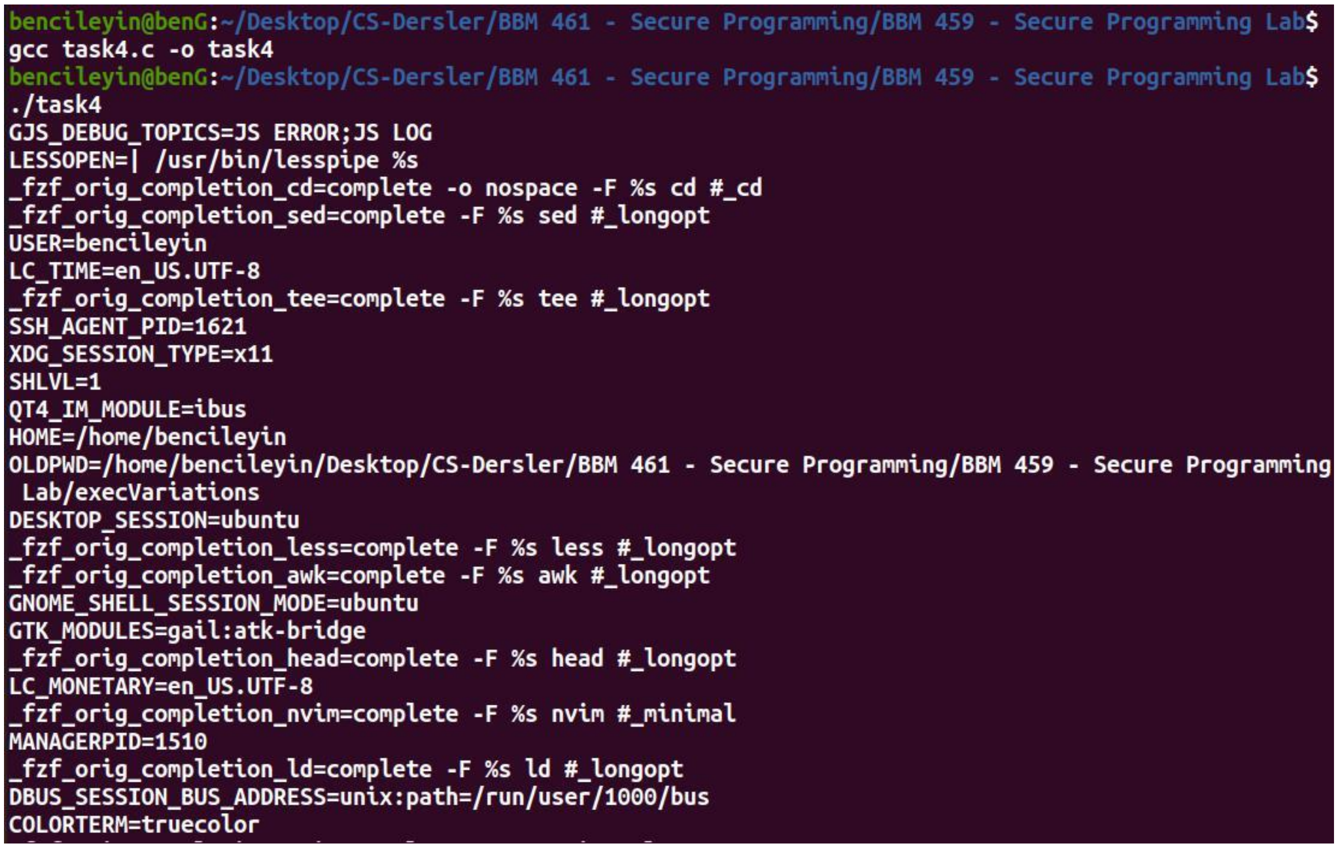
Task 4: Environment variables and "system()"

The given code is created as ***task4.c***



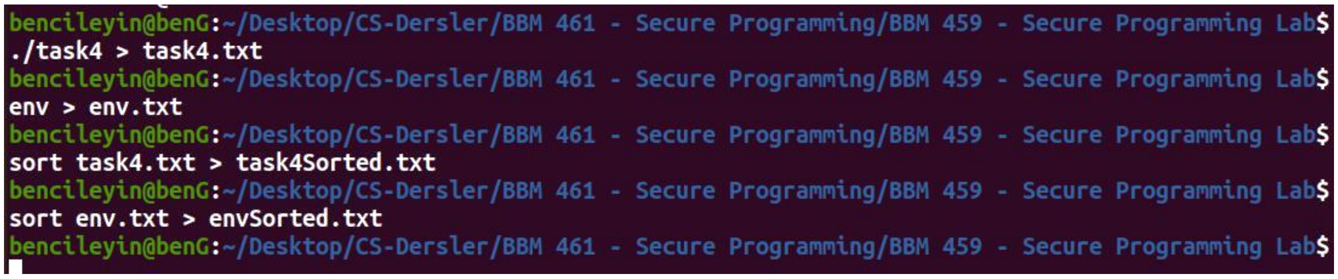
```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
  system("/usr/bin/env");
  return 0 ;
}
```

Afterwards, it is compiled into an executable called ***task4***. And the output is taken as follows:



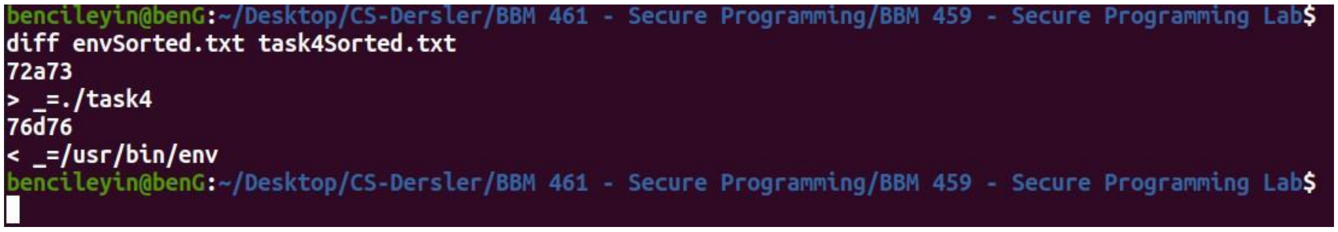
```
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
gcc task4.c -o task4
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
./task4
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
_fzf_orig_completion_cd=complete -o nospace -F %s cd #_cd
_fzf_orig_completion_sed=complete -F %s sed #_longopt
USER=bencileyin
LC_TIME=en_US.UTF-8
_fzf_orig_completion_tee=complete -F %s tee #_longopt
SSH_AGENT_PID=1621
XDG_SESSION_TYPE=x11
SHLVL=1
QT4_IM_MODULE=ibus
HOME=/home/bencileyin
OLDPWD=/home/bencileyin/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming
 Lab/execVariations
DESKTOP_SESSION=ubuntu
_fzf_orig_completion_less=complete -F %s less #_longopt
_fzf_orig_completion_awk=complete -F %s awk #_longopt
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
_fzf_orig_completion_head=complete -F %s head #_longopt
LC_MONETARY=en_US.UTF-8
_fzf_orig_completion_nvim=complete -F %s nvim #_minimal
MANAGERPID=1510
_fzf_orig_completion_ld=complete -F %s ld #_longopt
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
COLORTERM=truecolor
```

Output was messy. Before comparing with ***env*** output, both outputs are sorted with ***sort*** command.



```
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
./task4 > task4.txt
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
env > env.txt
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
sort task4.txt > task4Sorted.txt
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
sort env.txt > envSorted.txt
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
```

Lastly, ***envSorted.txt*** and ***task4Sorted.txt*** are compared with the help of ***diff*** command.



```
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
diff envSorted.txt task4Sorted.txt
72a73
> _=./task4
76d76
< _=/usr/bin/env
bencileyin@benG:~/Desktop/CS-Dersler/BBM 461 - Secure Programming/BBM 459 - Secure Programming Lab$
```

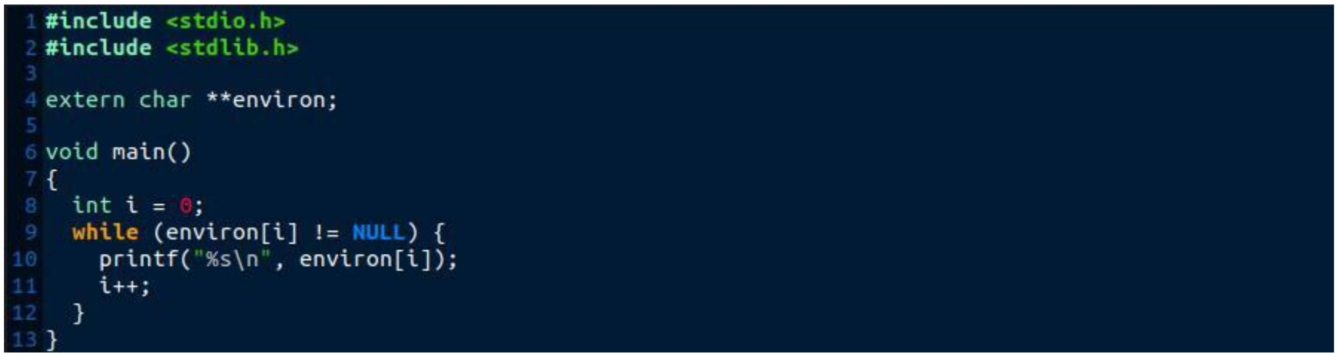
It is clear that the content of the environment variables in the shell are same as the output of the process which uses system method. The only difference is underscore variable, but it is not an environment variable but only a variable that is created by bash. Therefore we can safely conclude that system function call takes and pass the environment variables to the new process without any change.

Task 5: Environment variable and Set-UID Programs

bbm459user is created for task 5. This user is not a member of sudo group and does not have root privileges.

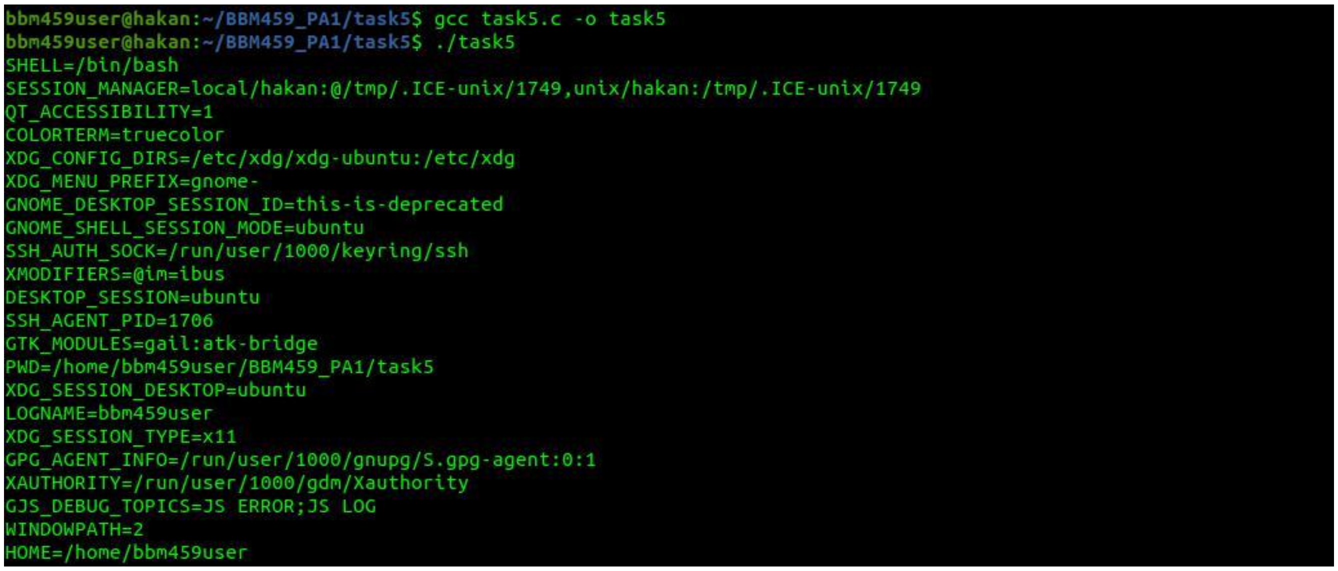
Step 1:

The given code is created as ***task5.c***



```c
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

void main()
{
  int i = 0;
  while (environ[i] != NULL) {
    printf("%s\n", environ[i]);
    i++;
  }
}
```

Afterwards, it is compiled into an executable called ***task5.*** And the output is taken as follows:

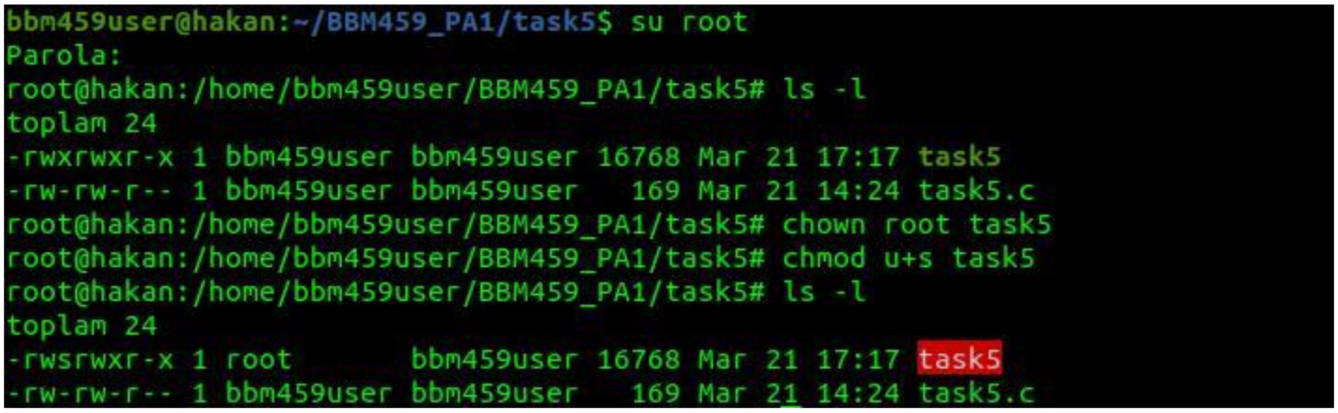


```
bbm459user@hakan:~/BBM459_PA1/task5$ gcc task5.c -o task5
bbm459user@hakan:~/BBM459_PA1/task5$ ./task5
SHELL=/bin/bash
SESSION_MANAGER=local/hakan:@/tmp/.ICE-unix/1749,unix/hakan:/tmp/.ICE-unix/1749
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1706
GTK_MODULES=gail:atk-bridge
PWD=/home/bbm459user/BBM459_PA1/task5
XDG_SESSION_DESKTOP=ubuntu
LOGNAME=bbm459user
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
HOME=/home/bbm459user
```

It can be seen that the environment variables have been printed out without any problems.

Step 2:

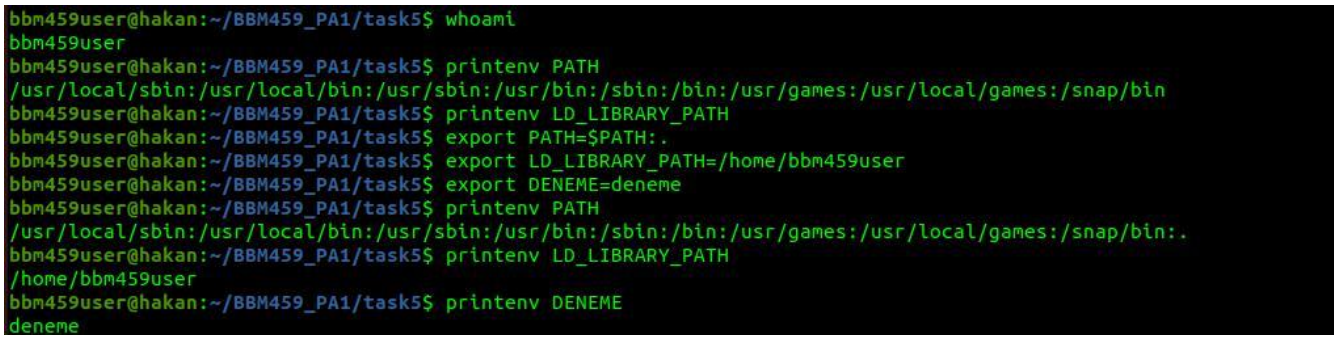
The owner of ***task5*** has been changed to root and the program is made as Set-UID Program:



```
bbm459user@hakan:~/BBM459_PA1/task5$ su root
Parola:
root@hakan:/home/bbm459user/BBM459_PA1/task5# ls -l
toplam 24
-rwxrwxr-x 1 bbm459user bbm459user 16768 Mar 21 17:17 task5
-rw-rw-r-- 1 bbm459user bbm459user   169 Mar 21 14:24 task5.c
root@hakan:/home/bbm459user/BBM459_PA1/task5# chown root task5
root@hakan:/home/bbm459user/BBM459_PA1/task5# chmod u+s task5
root@hakan:/home/bbm459user/BBM459_PA1/task5# ls -l
toplam 24
-rwsrwxr-x 1 root       bbm459user 16768 Mar 21 17:17 task5
-rw-rw-r-- 1 bbm459user bbm459user   169 Mar 21 14:24 task5.c
```

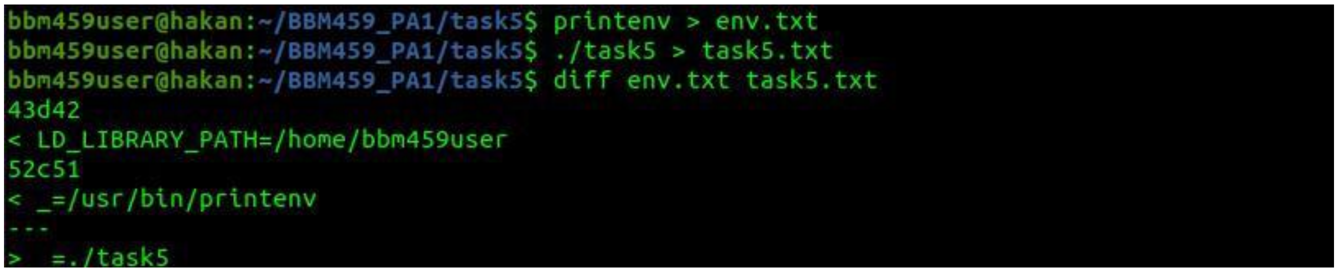
Step 3:

Additions are made to ***PATH*** and ***LD_LIBRARY_PATH*** variables and a new variable named ***DENEME*** is created:



```
bbm459user@hakan:~/BBM459_PA1/task5$ whoami
bbm459user
bbm459user@hakan:~/BBM459_PA1/task5$ printenv PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
bbm459user@hakan:~/BBM459_PA1/task5$ printenv LD_LIBRARY_PATH
bbm459user@hakan:~/BBM459_PA1/task5$ export PATH=$PATH:.
bbm459user@hakan:~/BBM459_PA1/task5$ export LD_LIBRARY_PATH=/home/bbm459user
bbm459user@hakan:~/BBM459_PA1/task5$ export DENEME=deneme
bbm459user@hakan:~/BBM459_PA1/task5$ printenv PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
bbm459user@hakan:~/BBM459_PA1/task5$ printenv LD_LIBRARY_PATH
/home/bbm459user
bbm459user@hakan:~/BBM459_PA1/task5$ printenv DENEME
deneme
```

All the environment variables is sets in the shell process (parent) must be checked. We must compare result of ***printenv*** command which will be executed in the shell process (parent) and the result of ***task5*** (child).



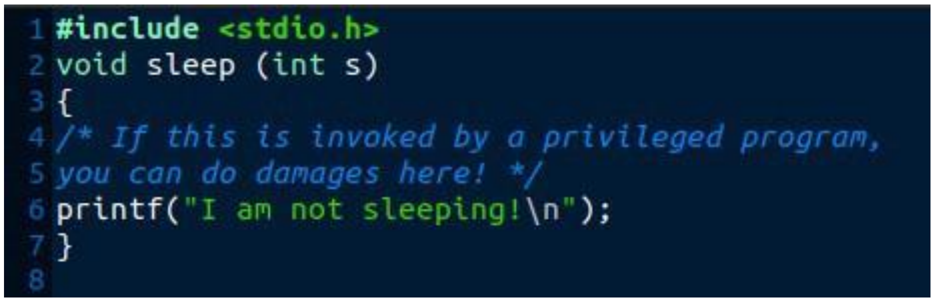
```
bbm459user@hakan:~/BBM459_PA1/task5$ printenv > env.txt
bbm459user@hakan:~/BBM459_PA1/task5$ ./task5 > task5.txt
bbm459user@hakan:~/BBM459_PA1/task5$ diff env.txt task5.txt
43d42
< LD_LIBRARY_PATH=/home/bbm459user
52c51
< _=/usr/bin/printenv
---
> _=./task5
```

PATH and ***DENEME*** are inherited but ***LD_LIBRARY_PATH*** is not inherited to child process. After some research, it is found that parent Set-UID program cannot keep the ***LD_LIBRARY_PATH*** as a part of the environment variables for security reasons. Because of that, any child process will not access the ***LD_LIBRARY_PATH***.

Task 6: The "LD_PRELOAD" environment variable and Set-UID Programs

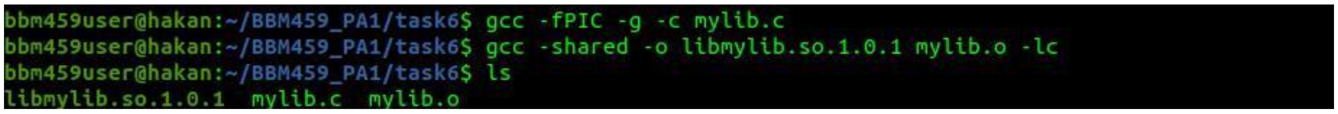
Step 1:

The given code is created as ***mylib.c:***



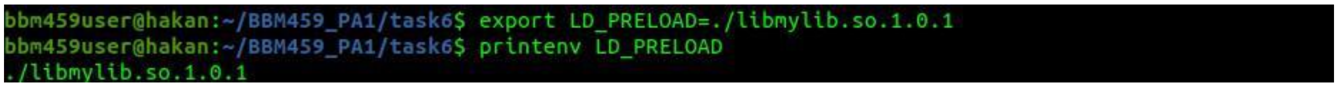
```c
#include <stdio.h>
void sleep (int s)
{
/* If this is invoked by a privileged program,
you can do damages here! */
printf("I am not sleeping!\n");
}

```

It is compiled using the following commands:



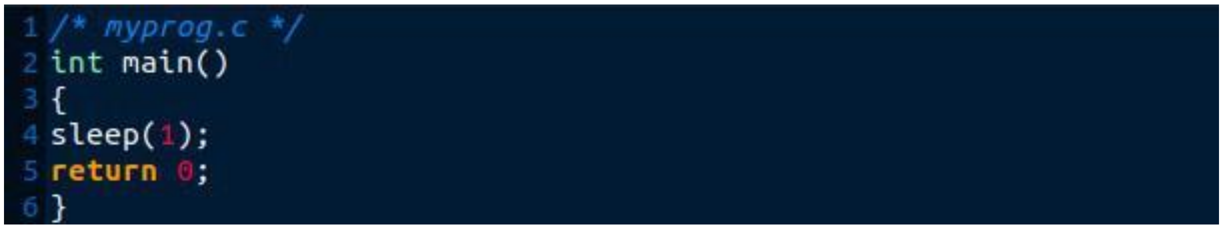
```
bbm459user@hakan:~/BBM459_PA1/task6$ gcc -fPIC -g -c mylib.c
bbm459user@hakan:~/BBM459_PA1/task6$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
bbm459user@hakan:~/BBM459_PA1/task6$ ls
libmylib.so.1.0.1  mylib.c  mylib.o
```

LD_PRELOAD environment variable is sets:



```
bbm459user@hakan:~/BBM459_PA1/task6$ export LD_PRELOAD=./libmylib.so.1.0.1
bbm459user@hakan:~/BBM459_PA1/task6$ printenv LD_PRELOAD
./libmylib.so.1.0.1
```

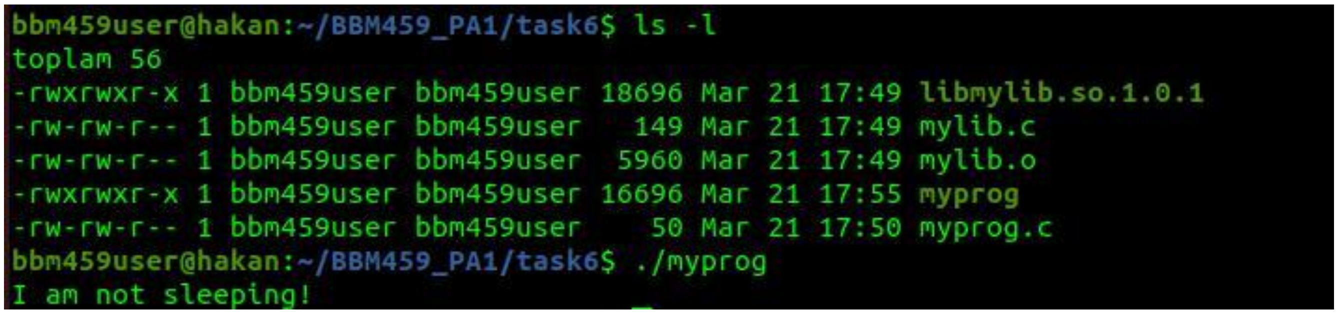
The given code is created as ***myprog.c:***



```c
/* myprog.c */
int main()
{
sleep(1);
return 0;
}
```

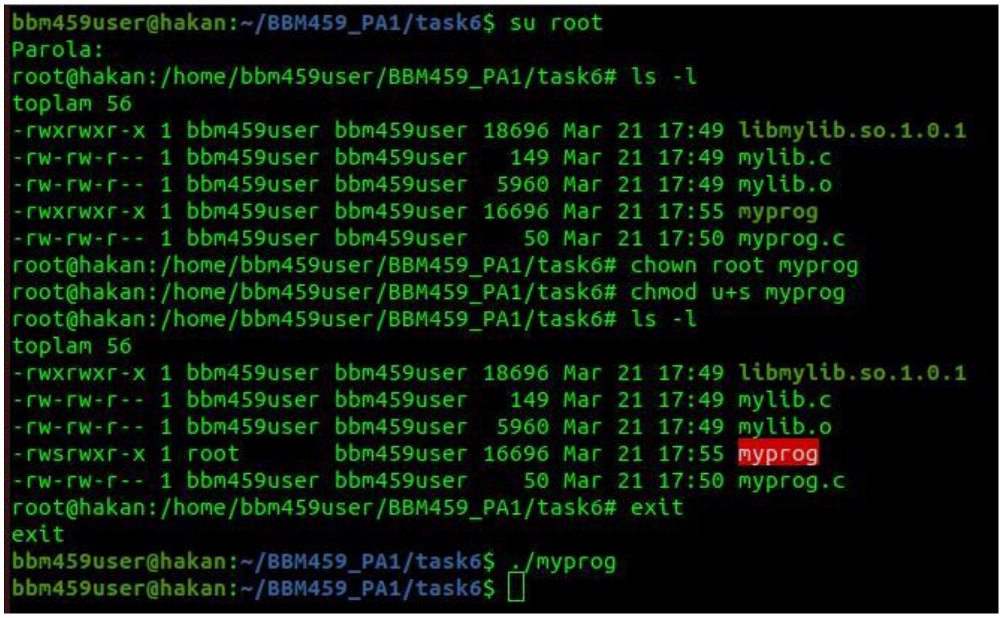
Step 2:

myprog is a regular program, runned as a normal user:



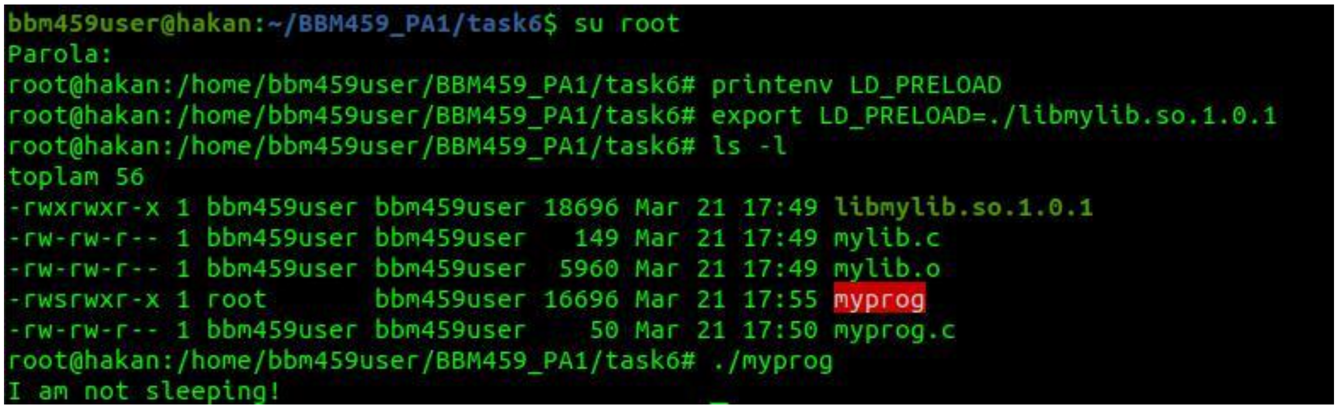
```
bbm459user@hakan:~/BBM459_PA1/task6$ ls -l
toplam 56
-rwxrwxr-x 1 bbm459user bbm459user 18696 Mar 21 17:49 libmylib.so.1.0.1
-rw-rw-r-- 1 bbm459user bbm459user   149 Mar 21 17:49 mylib.c
-rw-rw-r-- 1 bbm459user bbm459user  5960 Mar 21 17:49 mylib.o
-rwxrwxr-x 1 bbm459user bbm459user 16696 Mar 21 17:55 myprog
-rw-rw-r-- 1 bbm459user bbm459user    50 Mar 21 17:50 myprog.c
bbm459user@hakan:~/BBM459_PA1/task6$ ./myprog
I am not sleeping!
```

myprog is a Set-UID root program, runned as a normal user:



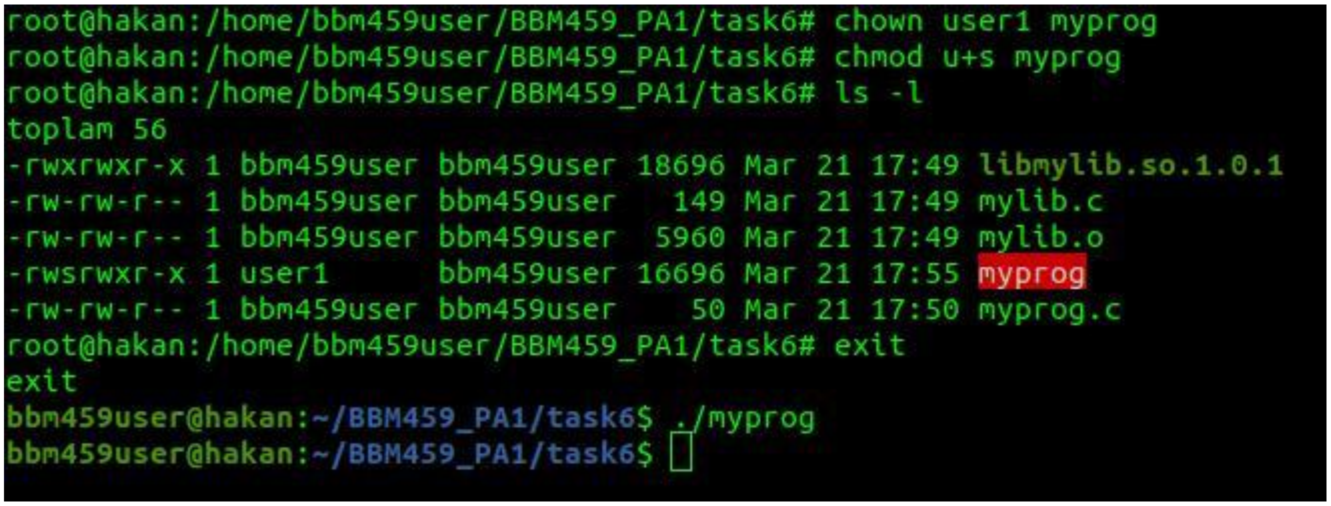
```
bbm459user@hakan:~/BBM459_PA1/task6$ su root
Parola:
root@hakan:/home/bbm459user/BBM459_PA1/task6# ls -l
toplam 56
-rwxrwxr-x 1 bbm459user bbm459user 18696 Mar 21 17:49 libmylib.so.1.0.1
-rw-rw-r-- 1 bbm459user bbm459user   149 Mar 21 17:49 mylib.c
-rw-rw-r-- 1 bbm459user bbm459user  5960 Mar 21 17:49 mylib.o
-rwxrwxr-x 1 bbm459user bbm459user 16696 Mar 21 17:55 myprog
-rw-rw-r-- 1 bbm459user bbm459user    50 Mar 21 17:50 myprog.c
root@hakan:/home/bbm459user/BBM459_PA1/task6# chown root myprog
root@hakan:/home/bbm459user/BBM459_PA1/task6# chmod u+s myprog
root@hakan:/home/bbm459user/BBM459_PA1/task6# ls -l
toplam 56
-rwxrwxr-x 1 bbm459user bbm459user 18696 Mar 21 17:49 libmylib.so.1.0.1
-rw-rw-r-- 1 bbm459user bbm459user   149 Mar 21 17:49 mylib.c
-rw-rw-r-- 1 bbm459user bbm459user  5960 Mar 21 17:49 mylib.o
-rwsrwxr-x 1 root       bbm459user 16696 Mar 21 17:55 myprog
-rw-rw-r-- 1 bbm459user bbm459user    50 Mar 21 17:50 myprog.c
root@hakan:/home/bbm459user/BBM459_PA1/task6# exit
exit
bbm459user@hakan:~/BBM459_PA1/task6$ ./myprog
bbm459user@hakan:~/BBM459_PA1/task6$ 
```

myprog is a Set-UID root program, ***LD_PRELOAD*** is exported in root and runned as root.



```
bbm459user@hakan:~/BBM459_PA1/task6$ su root
Parola:
root@hakan:/home/bbm459user/BBM459_PA1/task6# printenv LD_PRELOAD
root@hakan:/home/bbm459user/BBM459_PA1/task6# export LD_PRELOAD=./libmylib.so.1.0.1
root@hakan:/home/bbm459user/BBM459_PA1/task6# ls -l
toplam 56
-rwxrwxr-x 1 bbm459user bbm459user 18696 Mar 21 17:49 libmylib.so.1.0.1
-rw-rw-r-- 1 bbm459user bbm459user   149 Mar 21 17:49 mylib.c
-rw-rw-r-- 1 bbm459user bbm459user  5960 Mar 21 17:49 mylib.o
-rwsrwxr-x 1 root       bbm459user 16696 Mar 21 17:55 myprog
-rw-rw-r-- 1 bbm459user bbm459user    50 Mar 21 17:50 myprog.c
root@hakan:/home/bbm459user/BBM459_PA1/task6# ./myprog
I am not sleeping!
```

"***myprog***" is a Set-UID user1 program (the owner is user1, which is another user account), ***LD_PRELOAD*** environment variable is exported again in a different user's account (not-root user:bbm459user) and runned.



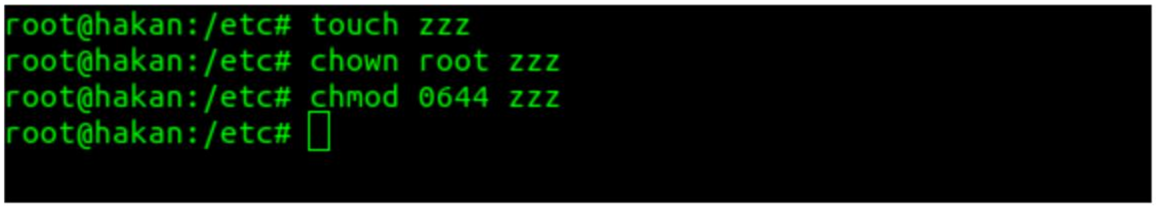
```
root@hakan:/home/bbm459user/BBM459_PA1/task6# chown user1 myprog
root@hakan:/home/bbm459user/BBM459_PA1/task6# chmod u+s myprog
root@hakan:/home/bbm459user/BBM459_PA1/task6# ls -l
toplam 56
-rwxrwxr-x 1 bbm459user bbm459user 18696 Mar 21 17:49 libmylib.so.1.0.1
-rw-rw-r-- 1 bbm459user bbm459user   149 Mar 21 17:49 mylib.c
-rw-rw-r-- 1 bbm459user bbm459user  5960 Mar 21 17:49 mylib.o
-rwsrwxr-x 1 user1      bbm459user 16696 Mar 21 17:55 myprog
-rw-rw-r-- 1 bbm459user bbm459user    50 Mar 21 17:50 myprog.c
root@hakan:/home/bbm459user/BBM459_PA1/task6# exit
exit
bbm459user@hakan:~/BBM459_PA1/task6$ ./myprog
bbm459user@hakan:~/BBM459_PA1/task6$ 
```

Step 3:

In the first and third case, when the real user ID and the effective user ID are the same, "***I am not sleeping!***" We encounter the phrase, so in these two cases, the variable ***LD_PRELOAD*** is inherited. However, in the second and fourth case, when the real user ID and effective user ID are different, our variable is not inherited. So the ***LD_PRELOAD*** variable (or all ***LD_ **** variables) is not inherited in Set-UID programs.

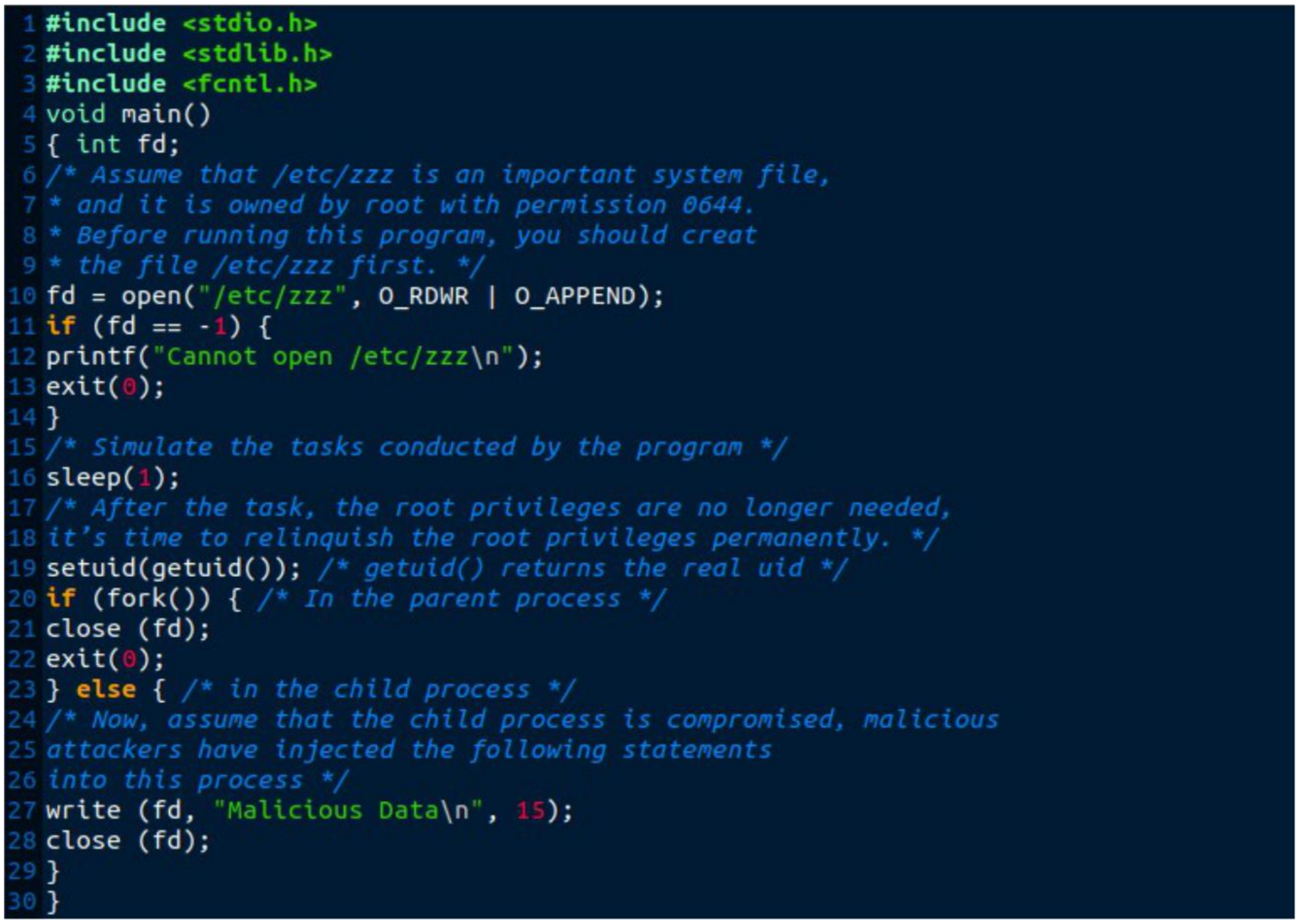
Task 7: Capability Leaking

zzz file is created, owner and permissions are changed:



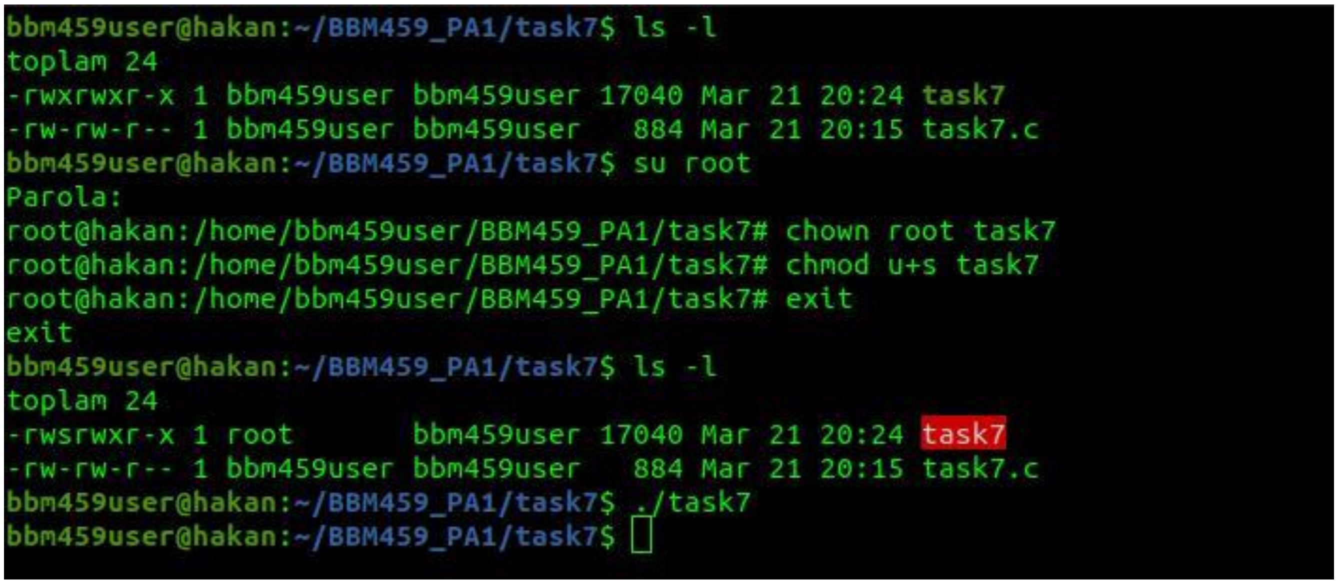
```
root@hakan:/etc# touch zzz
root@hakan:/etc# chown root zzz
root@hakan:/etc# chmod 0644 zzz
root@hakan:/etc# 
```

The given code is created as ***task7.c:***



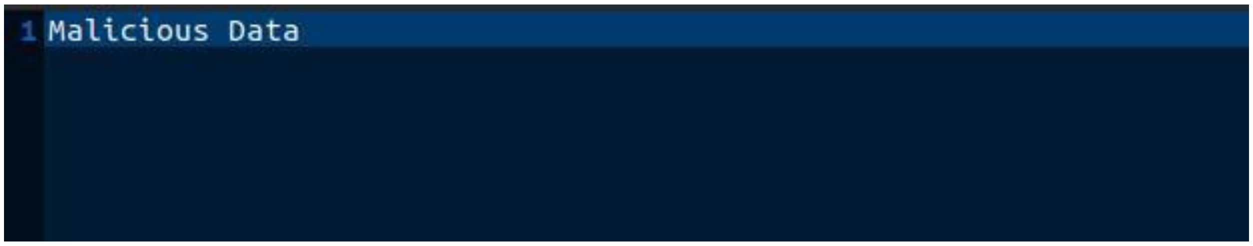
```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
void main()
{ int fd;
/* Assume that /etc/zzz is an important system file,
* and it is owned by root with permission 0644.
* Before running this program, you should creat
* the file /etc/zzz first. */
fd = open("/etc/zzz", O_RDWR | O_APPEND);
if (fd == -1) {
printf("Cannot open /etc/zzz\n");
exit(0);
}
/* Simulate the tasks conducted by the program */
sleep(1);
/* After the task, the root privileges are no longer needed,
it’s time to relinquish the root privileges permanently. */
setuid(getuid()); /* getuid() returns the real uid */
if (fork()) { /* In the parent process */
close (fd);
exit(0);
} else { /* in the child process */
/* Now, assume that the child process is compromised, malicious
attackers have injected the following statements
into this process */
write (fd, "Malicious Data\n", 15);
close (fd);
}
}
```

After that, ***task7*** is compiled and executable ***task7***’s owner changed as root. It is made a Set-UID program. When ***bbm459user***(non-root user) run it;



```
bbm459user@hakan:~/BBM459_PA1/task7$ ls -l
toplam 24
-rwxrwxr-x 1 bbm459user bbm459user 17040 Mar 21 20:24 task7
-rw-rw-r-- 1 bbm459user bbm459user   884 Mar 21 20:15 task7.c
bbm459user@hakan:~/BBM459_PA1/task7$ su root
Parola:
root@hakan:/home/bbm459user/BBM459_PA1/task7# chown root task7
root@hakan:/home/bbm459user/BBM459_PA1/task7# chmod u+s task7
root@hakan:/home/bbm459user/BBM459_PA1/task7# exit
exit
bbm459user@hakan:~/BBM459_PA1/task7$ ls -l
toplam 24
-rwsrwxr-x 1 root       bbm459user 17040 Mar 21 20:24 task7
-rw-rw-r-- 1 bbm459user bbm459user   884 Mar 21 20:15 task7.c
bbm459user@hakan:~/BBM459_PA1/task7$ ./task7
bbm459user@hakan:~/BBM459_PA1/task7$ 
```

Content of ***zzz*** is changed after execution:



```
Malicious Data
```

The content has changed because the parent process ended but the child process ran the malicious code before the process ended. The reason the malicious code is working is that the permissions are checked when the file is opened, not at every ***write()*** call. If the unprivileged shell is not supposed to write to the file, programmer must not leave the file open with the write access.