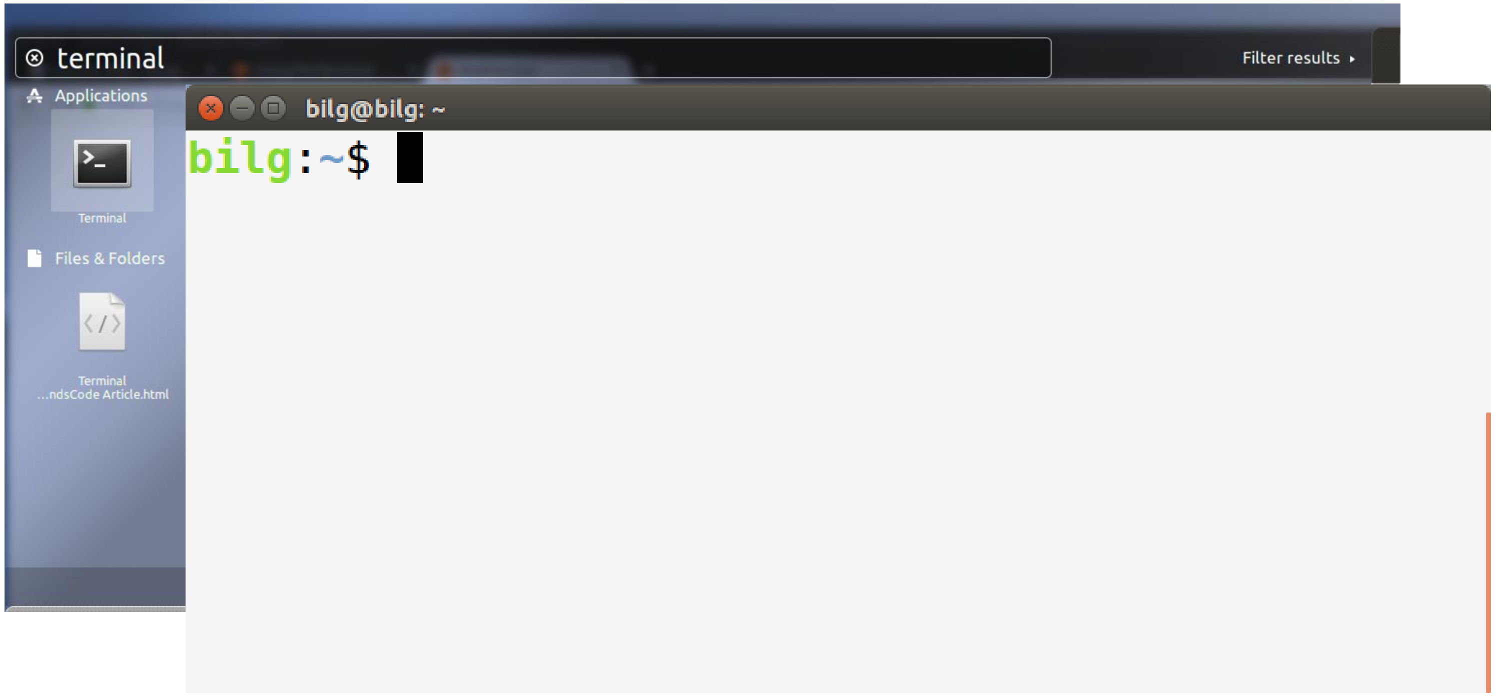
Terminal

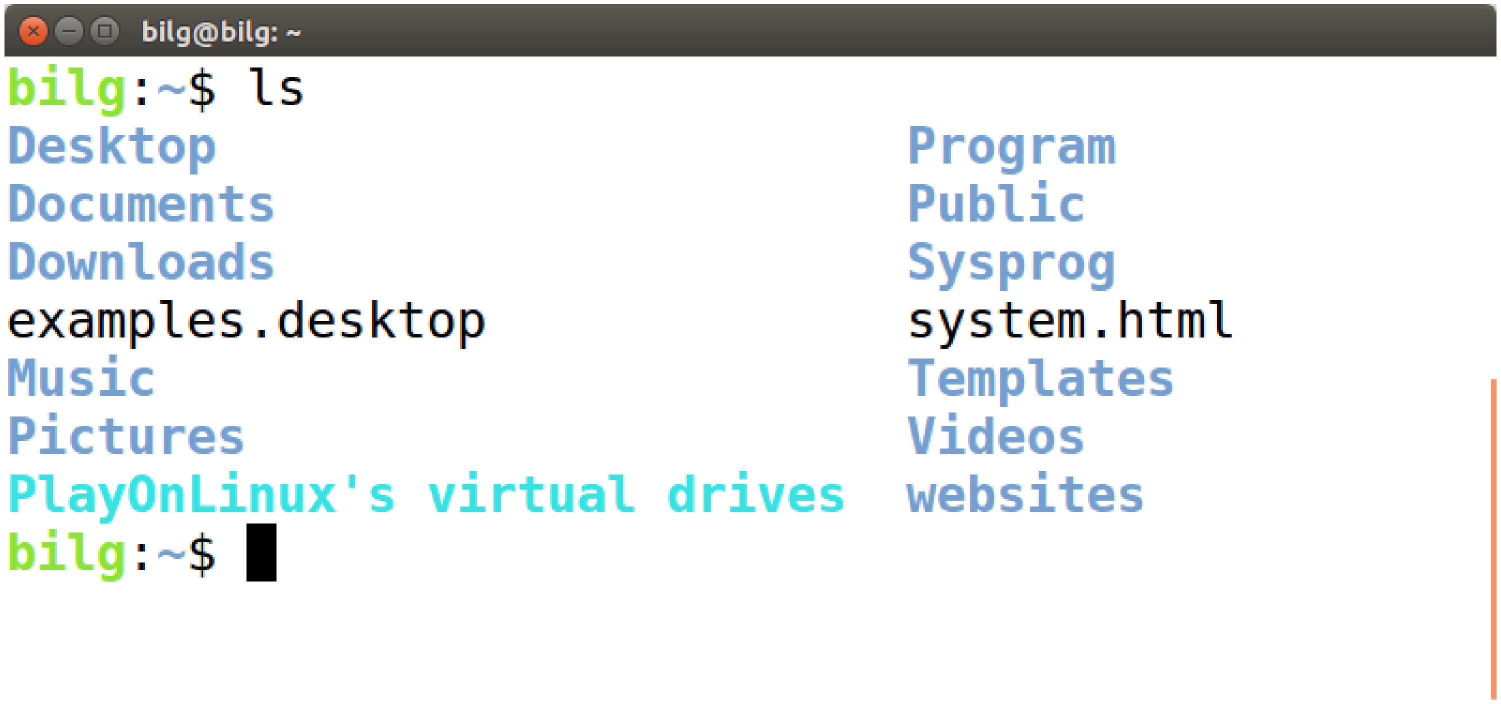
- We can issue commands to the operating system using the terminal screen, also called the shell.
- We can use Ctrl + Alt + T or program search menu to open the terminal screen.



```
bilg:~$
```

Terminal

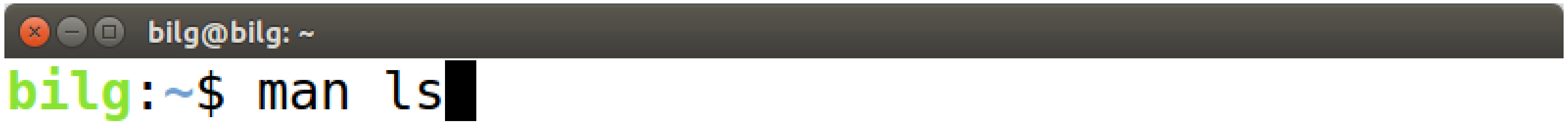
- If we run the ls command from the terminal screen, we can list the files in the active folder.



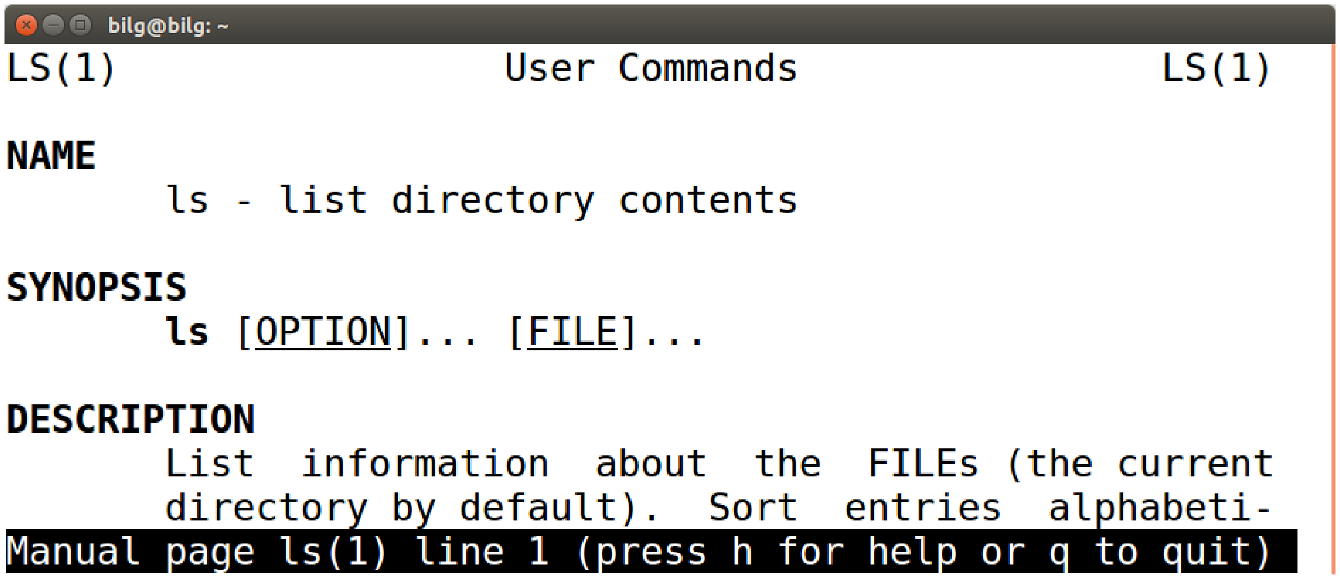
```
bilg:~$ ls
Desktop                          Program
Documents                        Public
Downloads                        Sysprog
examples.desktop                 system.html
Music                            Templates
Pictures                         Videos
PlayOnLinux's virtual drives     websites
bilg:~$ 
```

Terminal

- If we want to have information about a command we use on the terminal screen, we can use the man command.



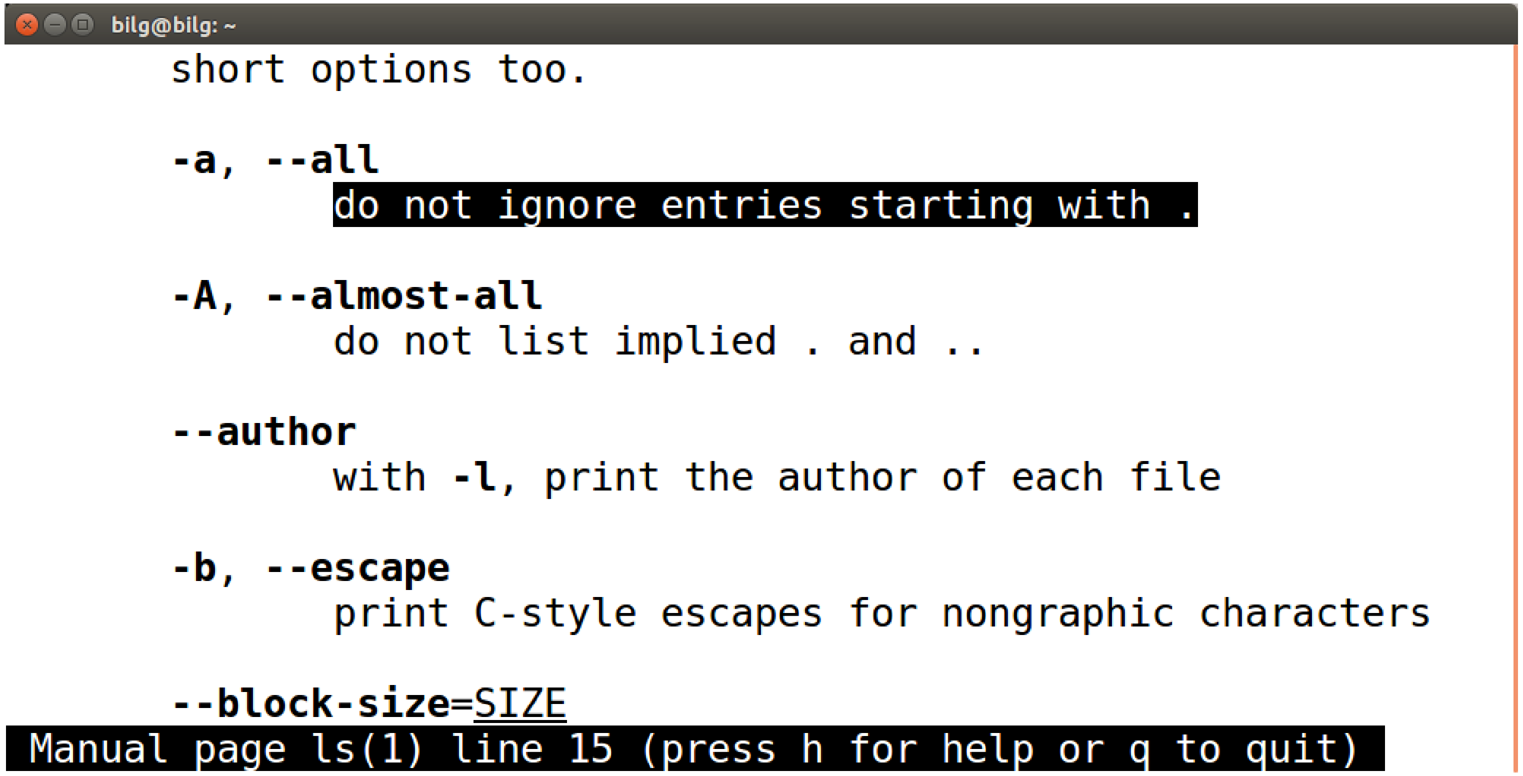
```
bilg:~$ man ls
```



```
LS(1)                    User Commands                    LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the  FILEs (the current
       directory by default).  Sort  entries  alphabeti-
Manual page ls(1) line 1 (press h for help or q to quit)
```

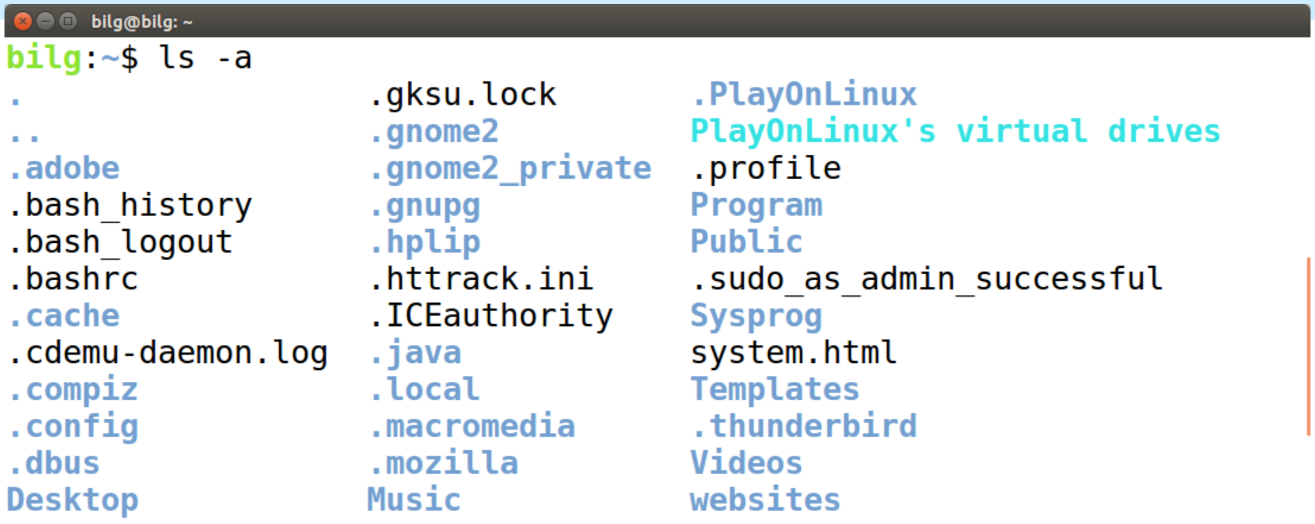
Terminal

- For example, if we use the -a option with ls. We can also view the records starting with.

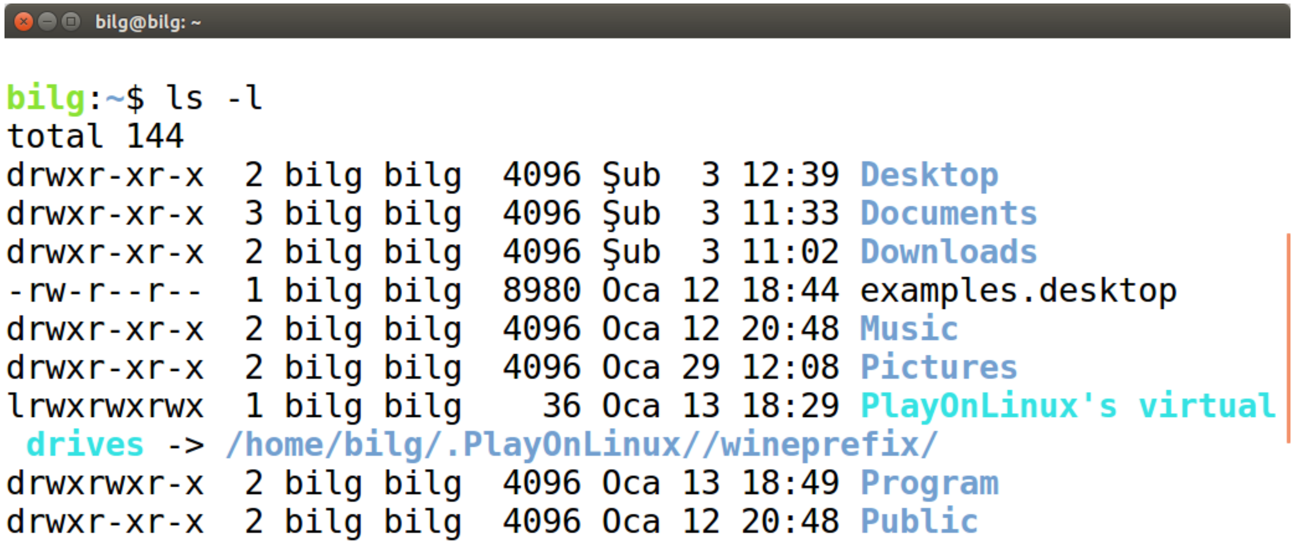


```
       short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

       --block-size=SIZE
 Manual page ls(1) line 15 (press h for help or q to quit)
```

Terminal



```
bilg@bilg: ~
bilg:~$ ls -a
.                    .gksu.lock          .PlayOnLinux
..                   .gnome2             PlayOnLinux's virtual drives
.adobe               .gnome2_private     .profile
.bash_history        .gnupg              Program
.bash_logout         .hplip              Public
.bashrc              .httrack.ini        .sudo_as_admin_successful
.cache               .ICEauthority       Sysprog
.cdemu-daemon.log    .java               system.html
.compiz              .local              Templates
.config              .macromedia         .thunderbird
.dbus                .mozilla            Videos
Desktop              Music               websites
```

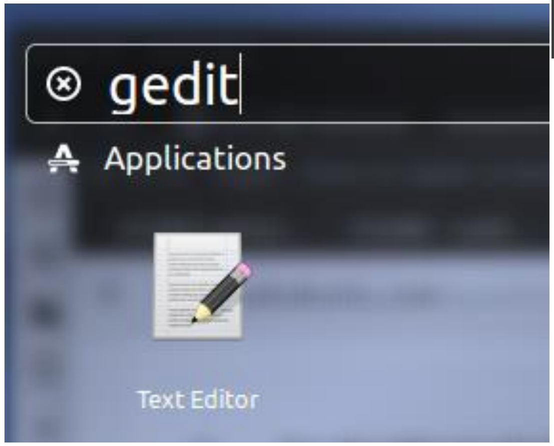


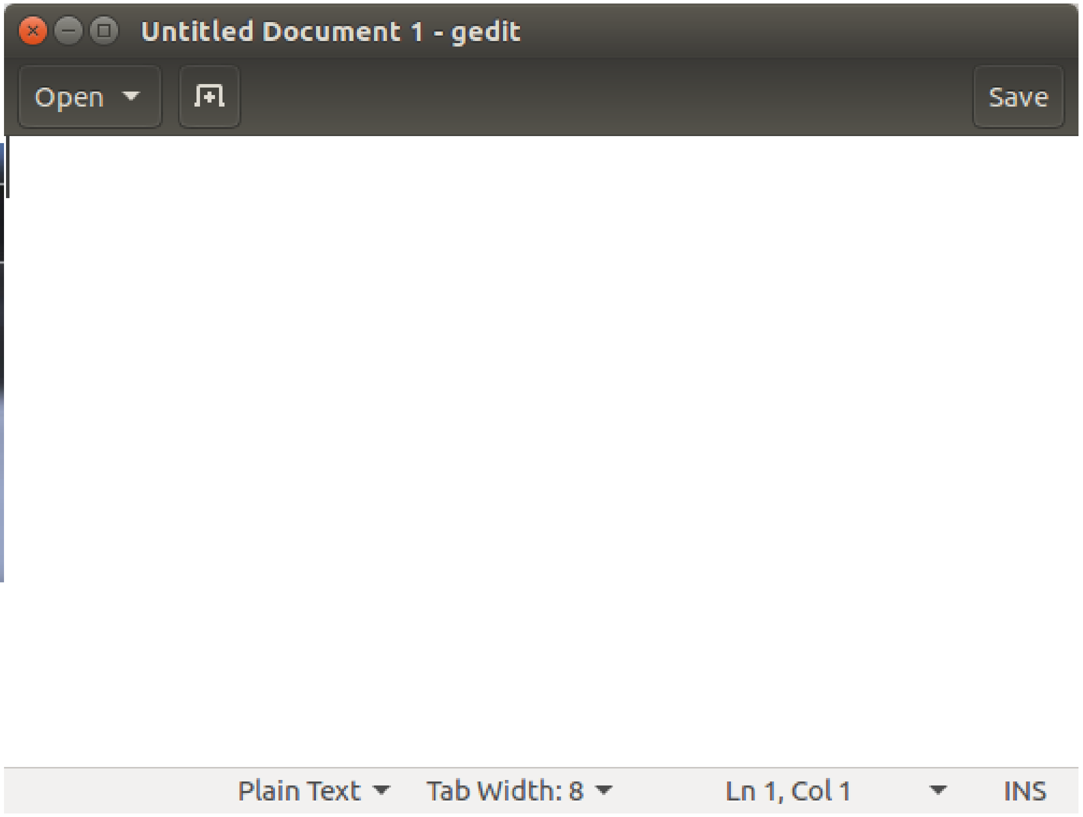
```
bilg@bilg: ~

bilg:~$ ls -l
total 144
drwxr-xr-x  2 bilg bilg  4096 Şub  3 12:39 Desktop
drwxr-xr-x  3 bilg bilg  4096 Şub  3 11:33 Documents
drwxr-xr-x  2 bilg bilg  4096 Şub  3 11:02 Downloads
-rw-r--r--  1 bilg bilg  8980 Oca 12 18:44 examples.desktop
drwxr-xr-x  2 bilg bilg  4096 Oca 12 20:48 Music
drwxr-xr-x  2 bilg bilg  4096 Oca 29 12:08 Pictures
lrwxrwxrwx  1 bilg bilg    36 Oca 13 18:29 PlayOnLinux's virtual
 drives -> /home/bilg/.PlayOnLinux//wineprefix/
drwxrwxr-x  2 bilg bilg  4096 Oca 13 18:49 Program
drwxr-xr-x  2 bilg bilg  4096 Oca 12 20:48 Public
```

ls -la

gedit

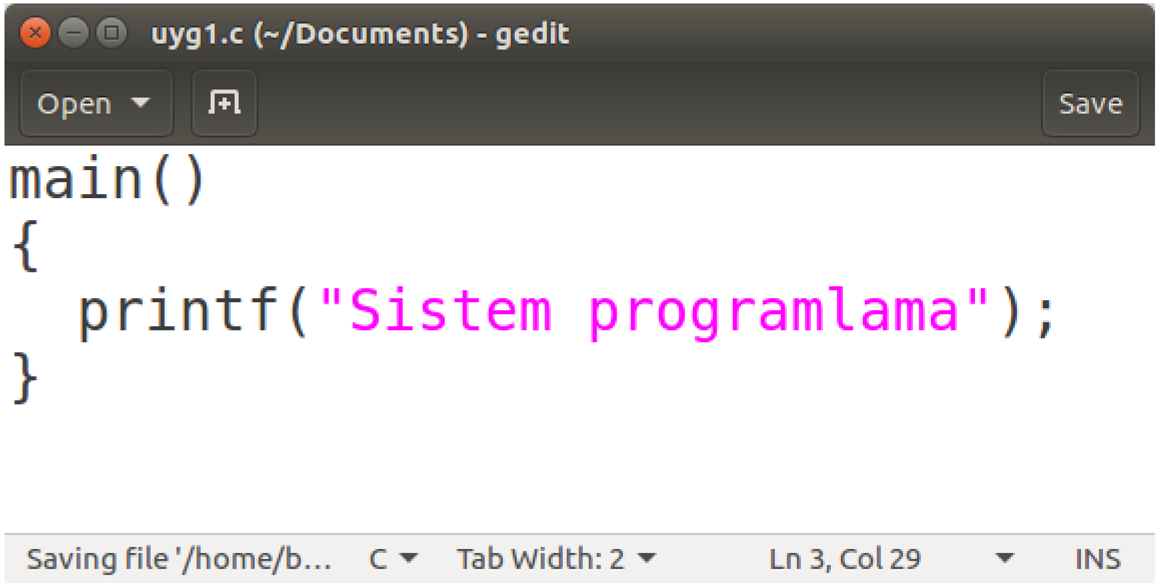
- ***gedit is an editor that comes built on ubuntu. We will use it when writing the C code.***





Example program in C

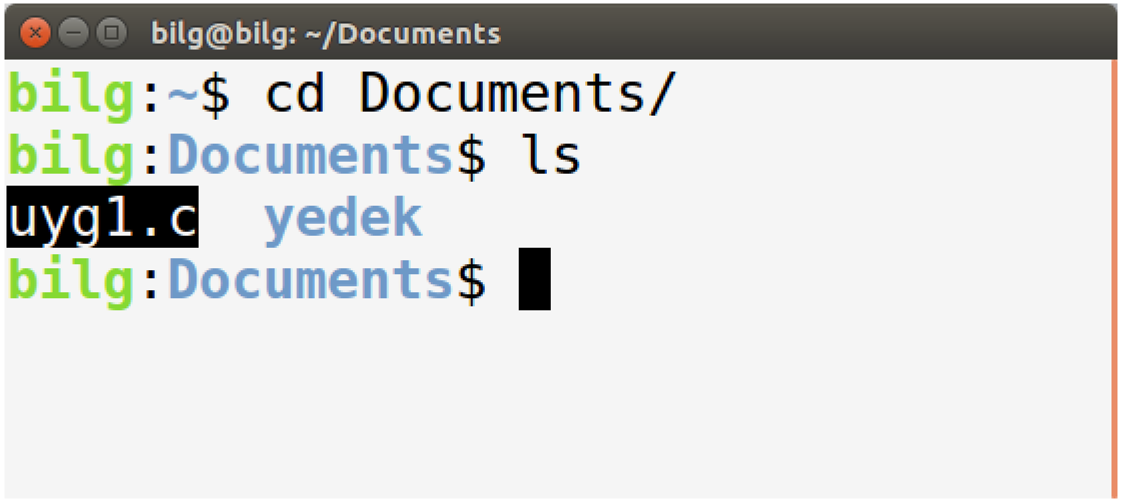
- Using the editor, a simple program can be written as follows.
- The program will only print a message on the screen.



```
main()
{
  printf("Sistem programlama");
}
```

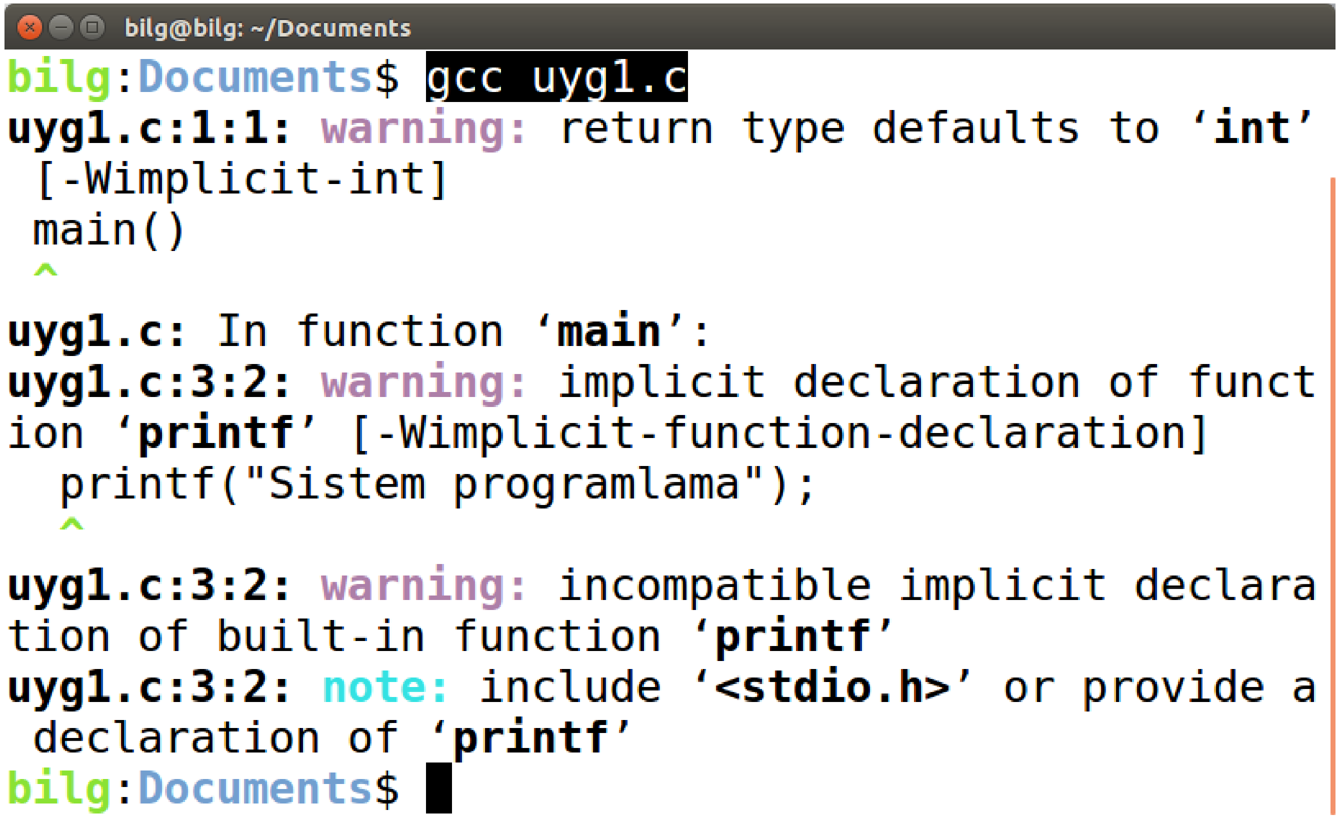
Compiling the program

- After saving the app1.c file, let's compile it using GNU C Compiler.
- First of all, let's change the directory we are in to be the folder where the apps1.c file is located.



```
bilg:~$ cd Documents/
bilg:Documents$ ls
uyg1.c   yedek
bilg:Documents$ 
```

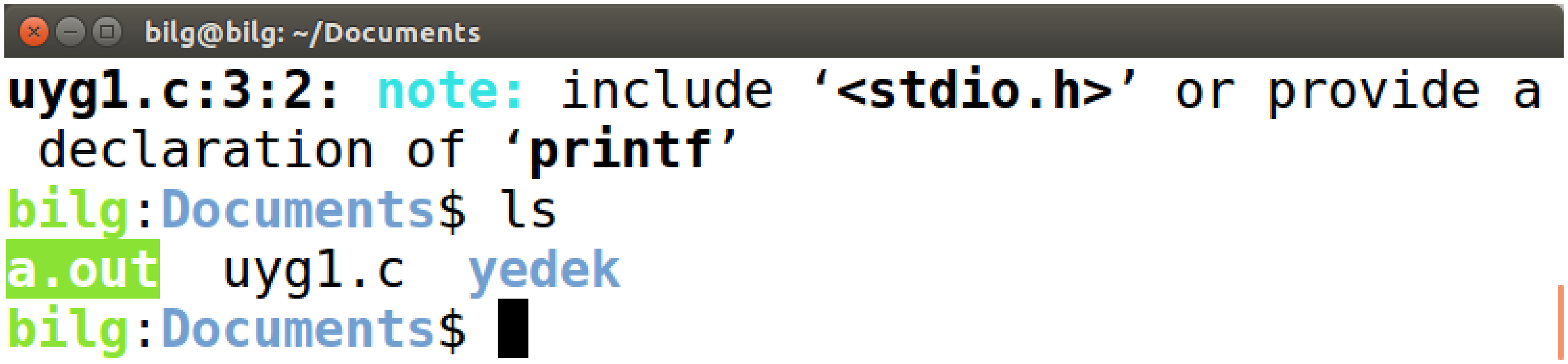
- After typing gcc as below, if we run by specifying the filename, our file will be compiled.
- The warnings that arise as a result of the compilation are about the construction of the libraries that we did not add.



```
bilg:Documents$ gcc uyg1.c
uyg1.c:1:1: warning: return type defaults to ‘int’
 [-Wimplicit-int]
 main()
 ^

uyg1.c: In function ‘main’:
uyg1.c:3:2: warning: implicit declaration of funct
ion ‘printf’ [-Wimplicit-function-declaration]
  printf("Sistem programlama");
  ^

uyg1.c:3:2: warning: incompatible implicit declara
tion of built-in function ‘printf’
uyg1.c:3:2: note: include ‘<stdio.h>’ or provide a
 declaration of ‘printf’
bilg:Documents$ 
```

Compiling the program

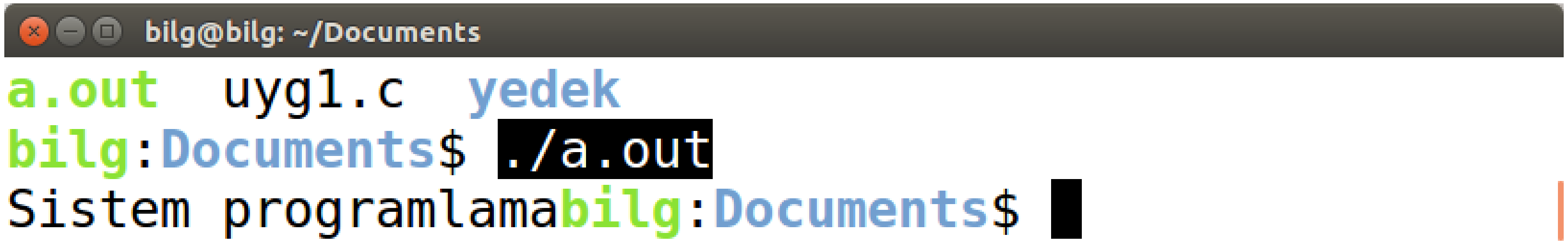
- If we type ls again from the command window and look at the files in the folder, a file named a.out is created as a result of the compilation.



```
uyg1.c:3:2: note: include ‘<stdio.h>’ or provide a
 declaration of ‘printf’
bilg:Documents$ ls
a.out  uyg1.c  yedek
bilg:Documents$ 
```

Running the program

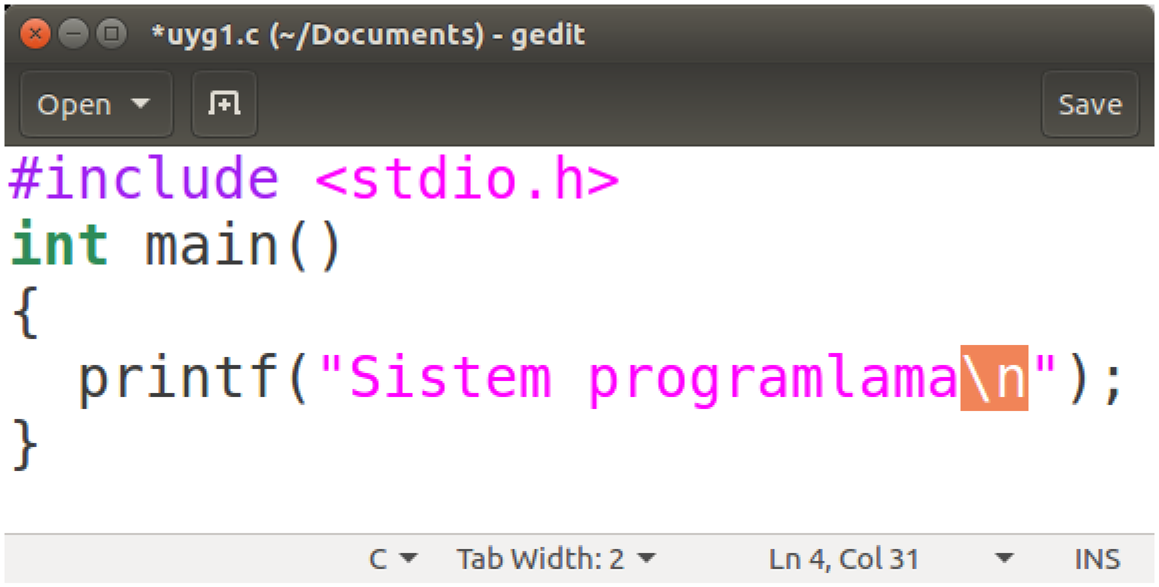
- If we write ./a.out on the command line, the program works as expected and prints a message on the screen.



```
a.out   uyg1.c   yedek
bilg:Documents$ ./a.out
Sistem programlamabilg:Documents$ 
```

Removing compilation warnings

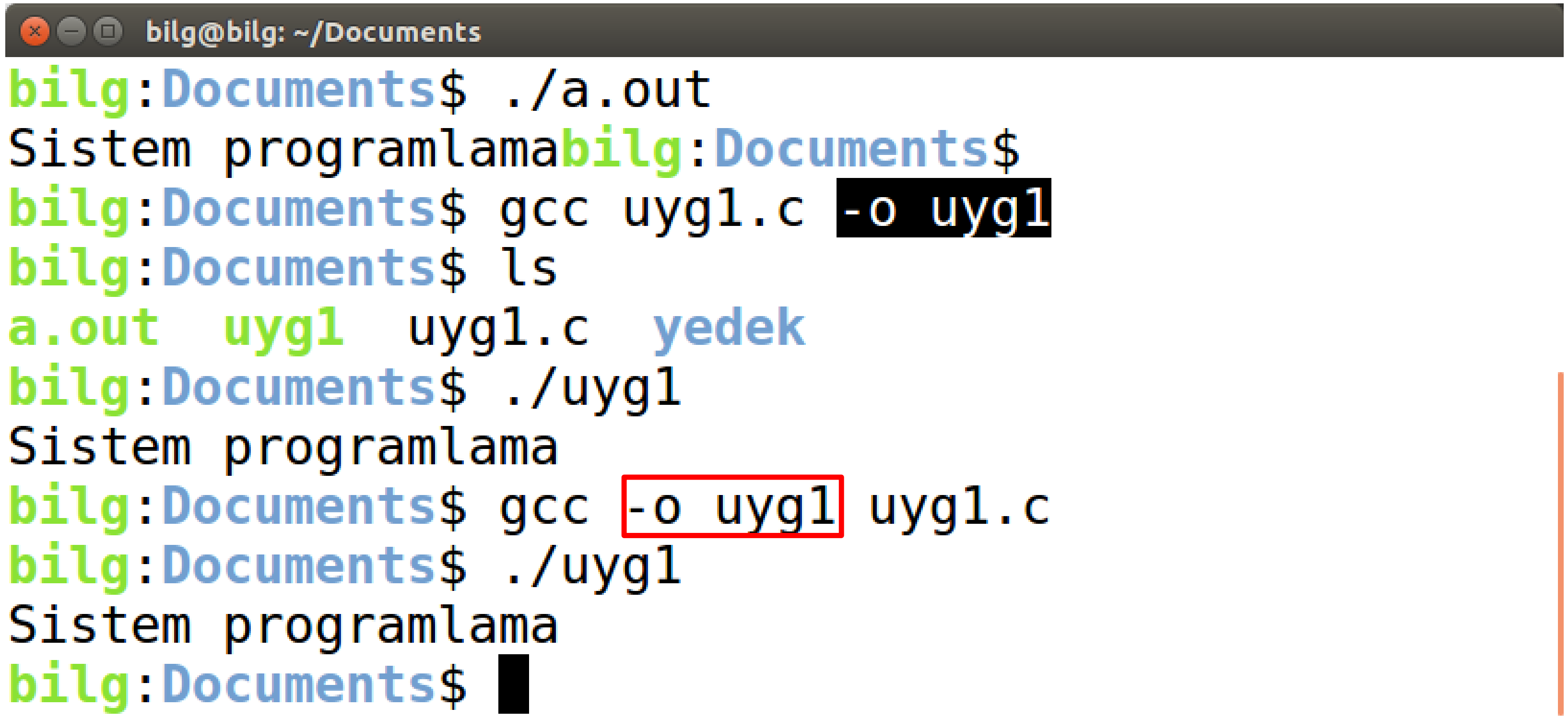
- To remove the compilation warnings, let's add the stdio.h library, which provides the printf function to the program, and make the return type of the main function int.



```
#include <stdio.h>
int main()
{
  printf("Sistem programlama\n");
}
```

Changing the compiled program name

- To change the name of the program we compiled, the –o option is used as follows:
- gcc test.c **-o programname**

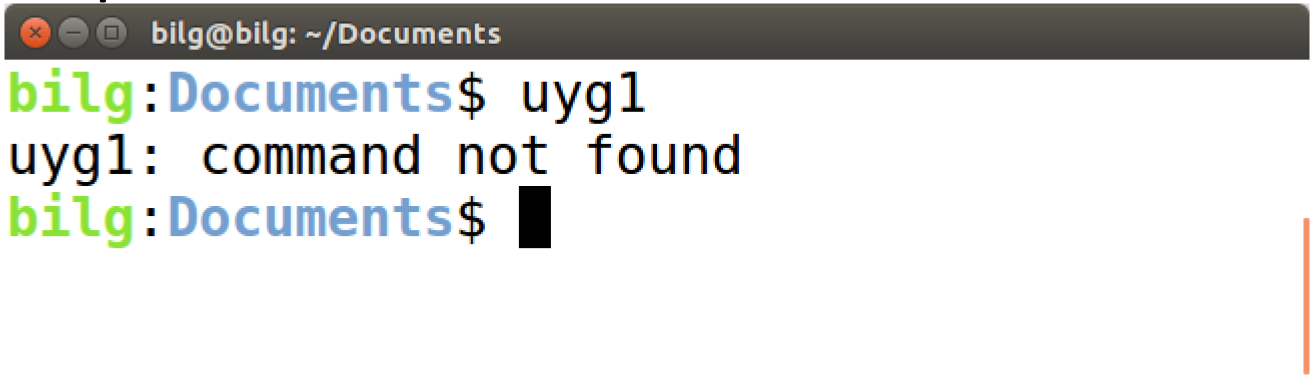


bilg@bilg: ~/Documents

```
bilg:Documents$ ./a.out
Sistem programlamabilg:Documents$
bilg:Documents$ gcc uyg1.c -o uyg1
bilg:Documents$ ls
a.out  uyg1  uyg1.c  yedek
bilg:Documents$ ./uyg1
Sistem programlama
bilg:Documents$ gcc -o uyg1 uyg1.c
bilg:Documents$ ./uyg1
Sistem programlama
bilg:Documents$ 
```

PATH variable

- When running the program, the compiled program that we write in front of the compiled file prevents it from being confused with the programs in the system path.

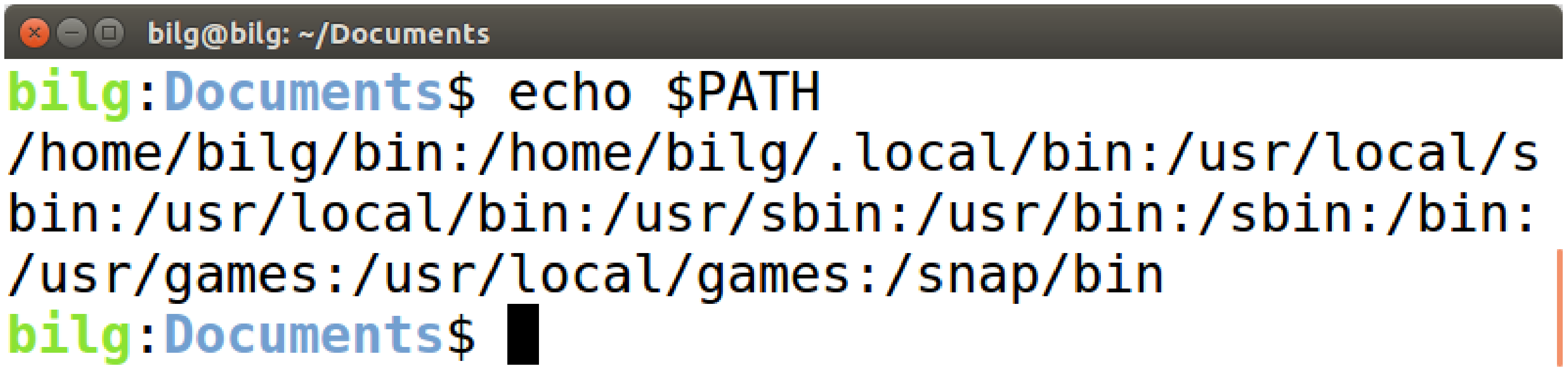


```
bilg:Documents$ uyg1
uyg1: command not found
bilg:Documents$ 
```

- If we want to run the program without using ./, we need to add the directory /folder containing the compiled code to the system path.

PATH

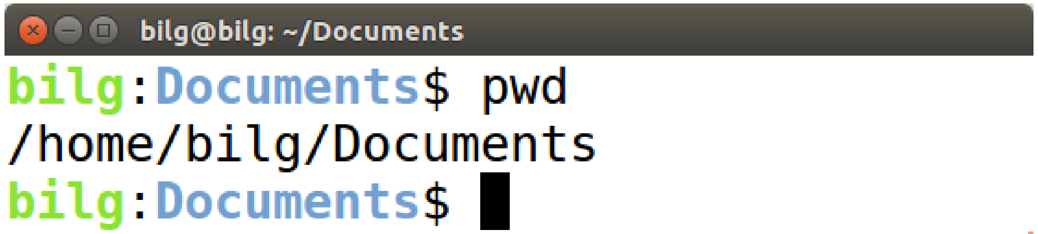
We can print the PATH variable with the echo command to see the directories in the system path.



```
bilg:Documents$ echo $PATH
/home/bilg/bin:/home/bilg/.local/bin:/usr/local/s
bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games:/usr/local/games:/snap/bin
bilg:Documents$ 
```

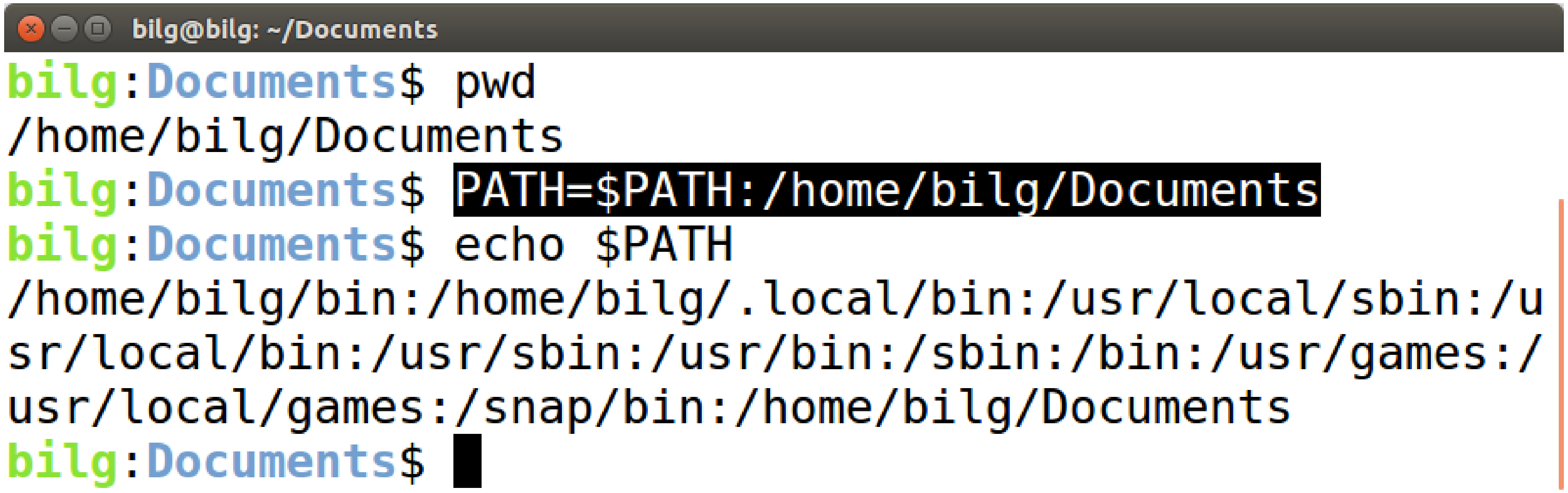
PATH variable

- We can use pwd (print working directory) command to see the path of the directory we are in.



```
bilg@bilg: ~/Documents
bilg:Documents$ pwd
/home/bilg/Documents
bilg:Documents$ 
```

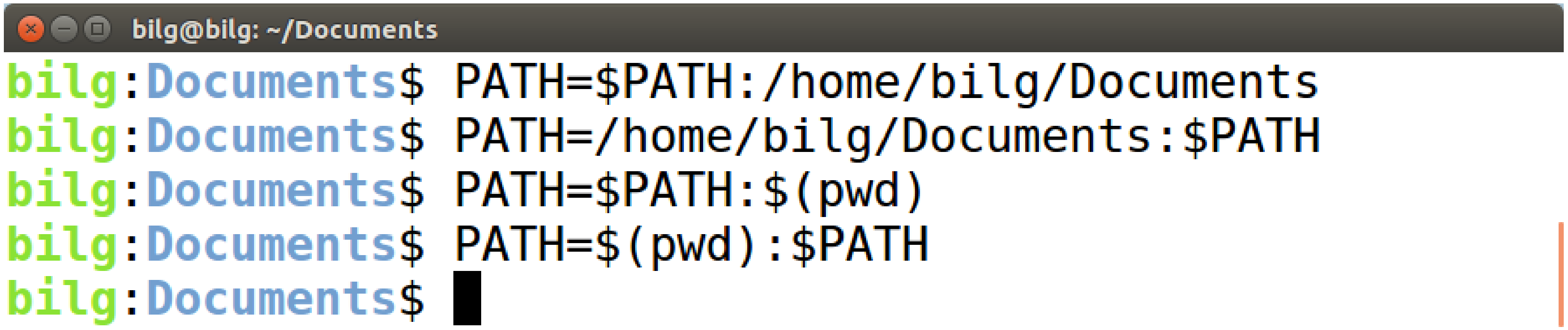
- Bu yolu kopyalayıp sistem yoluna ekleyebiliriz.



```
bilg@bilg: ~/Documents
bilg:Documents$ pwd
/home/bilg/Documents
bilg:Documents$ PATH=$PATH:/home/bilg/Documents
bilg:Documents$ echo $PATH
/home/bilg/bin:/home/bilg/.local/bin:/usr/local/sbin:/u
sr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/
usr/local/games:/snap/bin:/home/bilg/Documents
bilg:Documents$ 
```

PATH variable

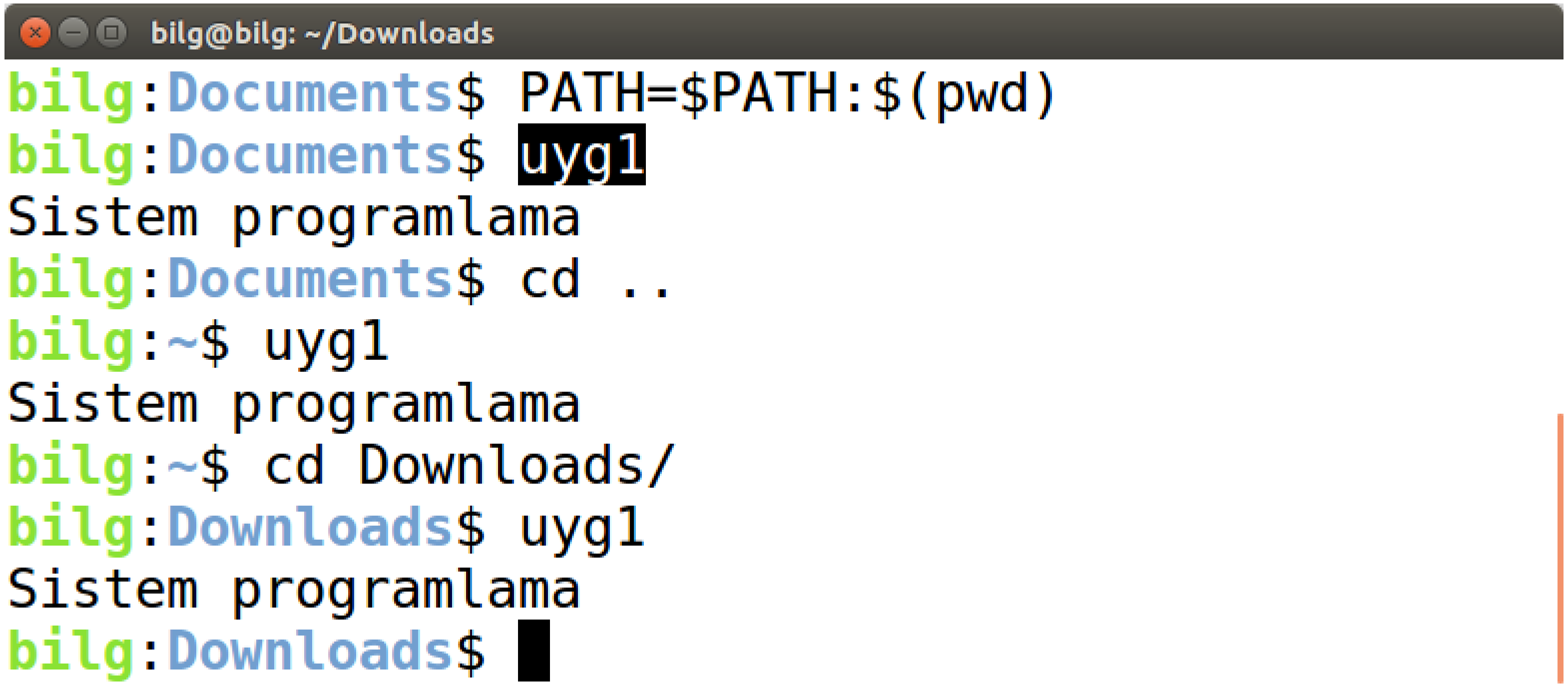
- Instead of writing the path, we can add the active directory to the system path directly with the pwd command.



```
bilg:Documents$ PATH=$PATH:/home/bilg/Documents
bilg:Documents$ PATH=/home/bilg/Documents:$PATH
bilg:Documents$ PATH=$PATH:$(pwd)
bilg:Documents$ PATH=$(pwd):$PATH
bilg:Documents$ 
```

PATH

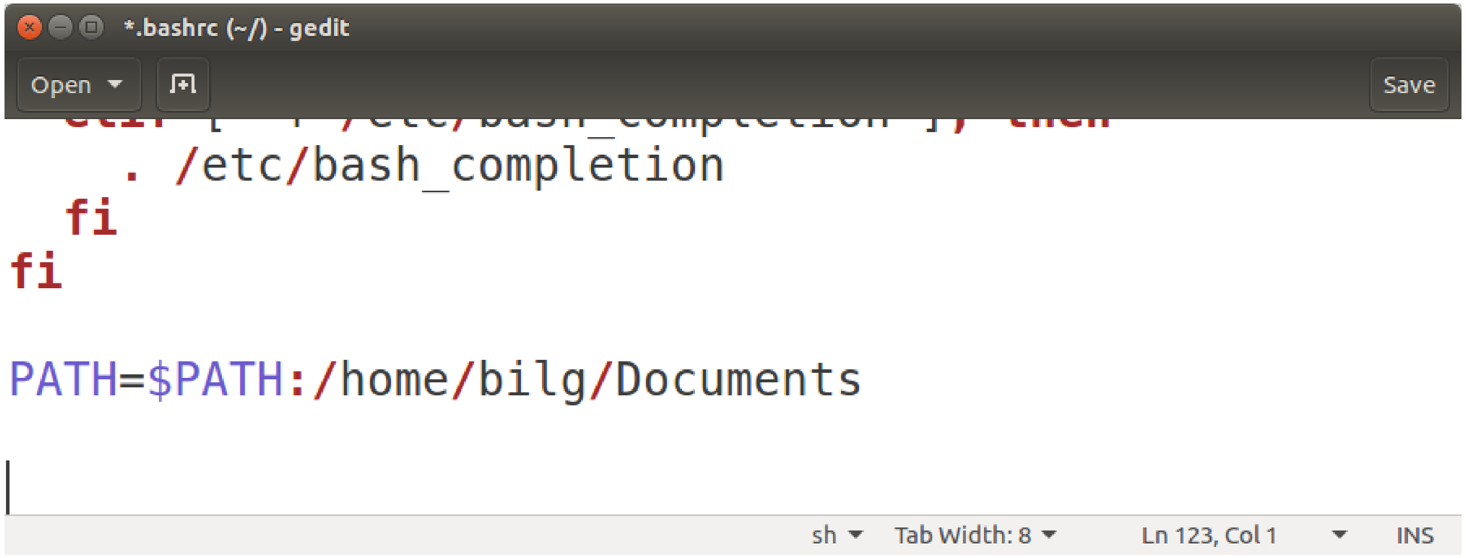
- When we add the program to the system path, we can write and run it directly without using ./.



```
bilg:Documents$ PATH=$PATH:$(pwd)
bilg:Documents$ uyg1
Sistem programlama
bilg:Documents$ cd ..
bilg:~$ uyg1
Sistem programlama
bilg:~$ cd Downloads/
bilg:Downloads$ uyg1
Sistem programlama
bilg:Downloads$ 
```

PATH

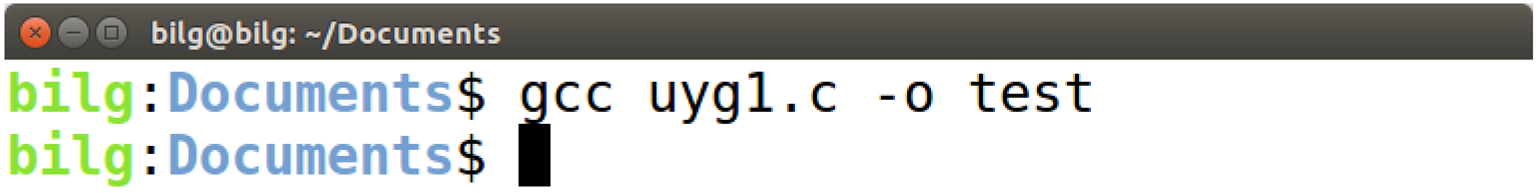
- Our changes are lost when we close and open the terminal window.
- We can use the ~ / .bashrc file to be permanent.
- From the command line: gedit ~ / .bashrc



```
    . /etc/bash_completion
  fi
fi

PATH=$PATH:/home/bilg/Documents

```

PATH

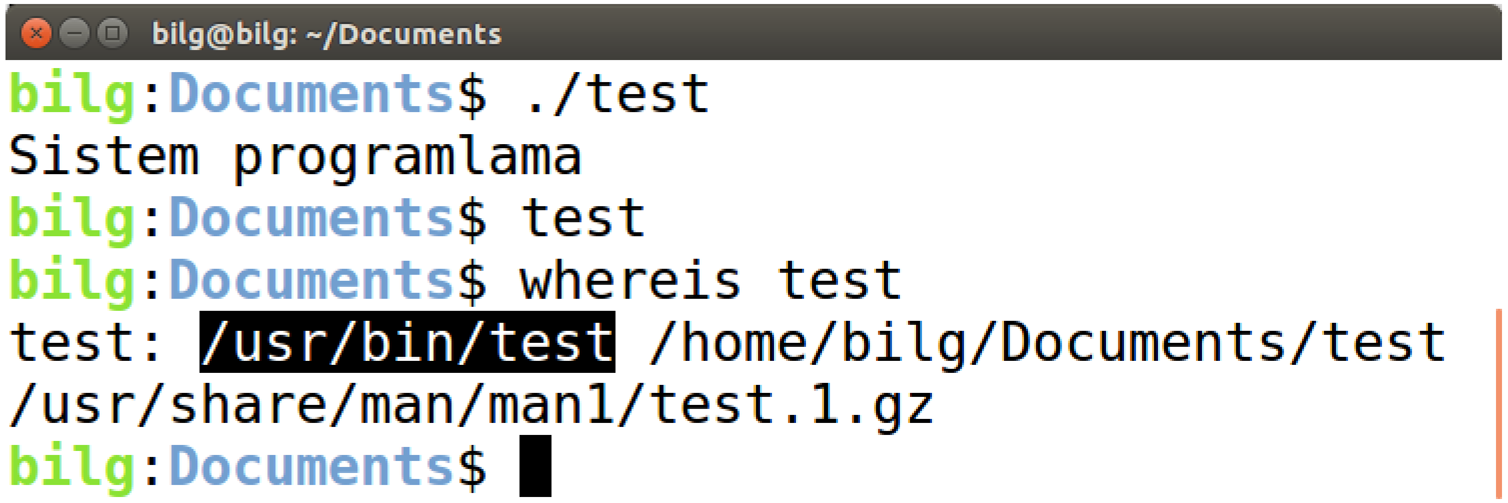
- It should be noted that the programs in the folders we add to the system path and the programs in the other folder should not have the same name.
- For example, let's compile the program we are evaluating by giving the test name instead of uyg1.



```
bilg:Documents$ gcc uyg1.c -o test
bilg:Documents$ 
```

whereis

- In this case, the program in another folder runs instead of our program.



bilg@bilg: ~/Documents

```
bilg:Documents$ ./test
Sistem programlama
bilg:Documents$ test
bilg:Documents$ whereis test
test: /usr/bin/test /home/bilg/Documents/test
/usr/share/man/man1/test.1.gz
bilg:Documents$ 
```