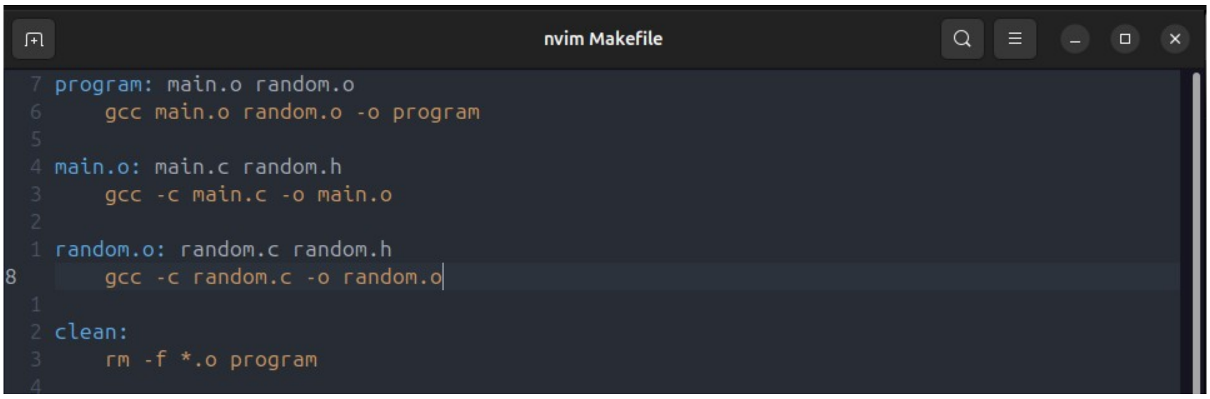
Embedded Linux Lab 3

Basic Make

Created the make file and wrote the recipe. But I didn't add .PHONY



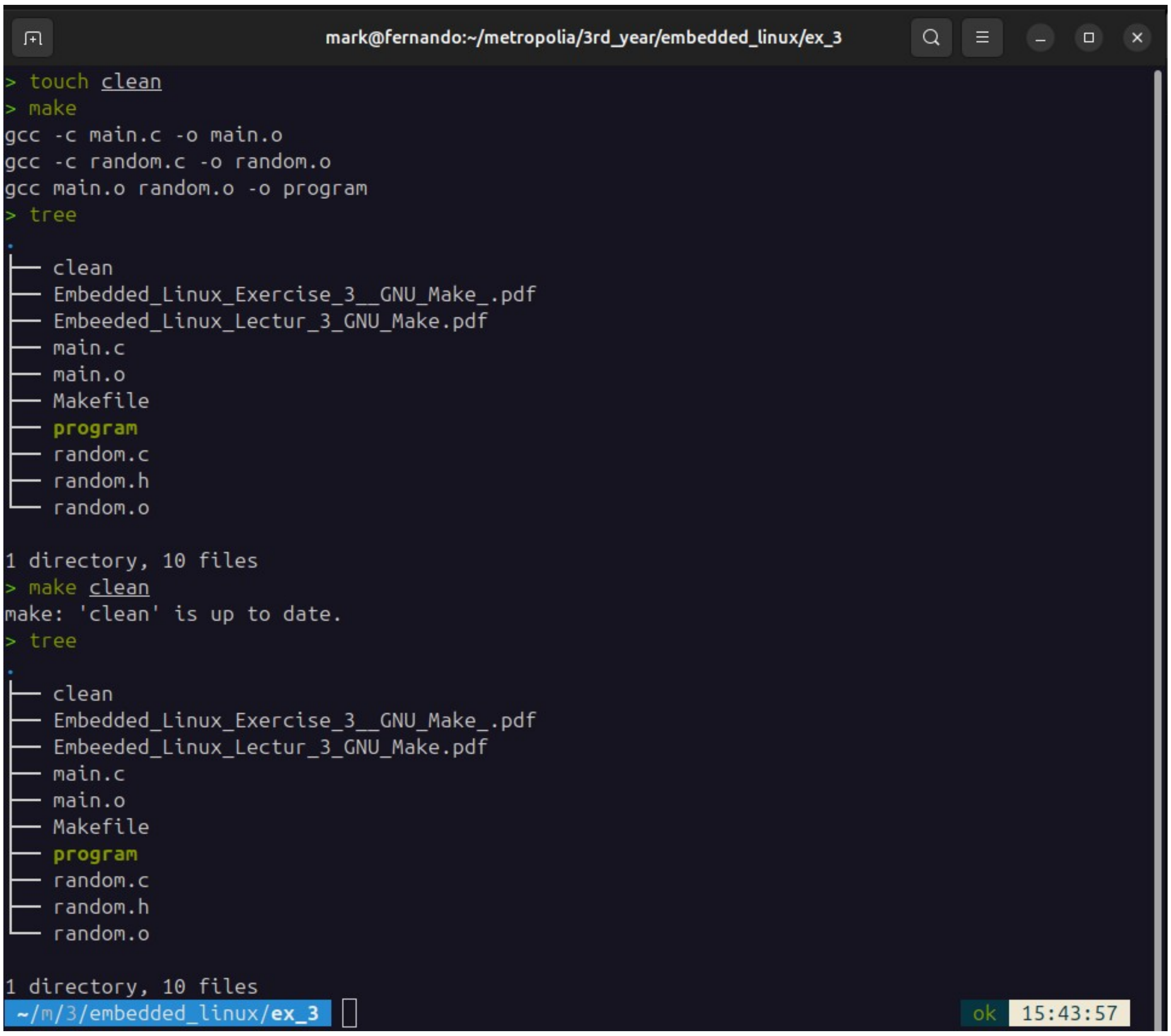
```
program: main.o random.o
	gcc main.o random.o -o program

main.o: main.c random.h
	gcc -c main.c -o main.o

random.o: random.c random.h
	gcc -c random.c -o random.o

clean:
	rm -f *.o program
```

Then I ran the program. I could see the object file in my directory. Then I created a file called clean and delete all object files and program file. Then again ran make. After that I ran make clean. Here is the output.

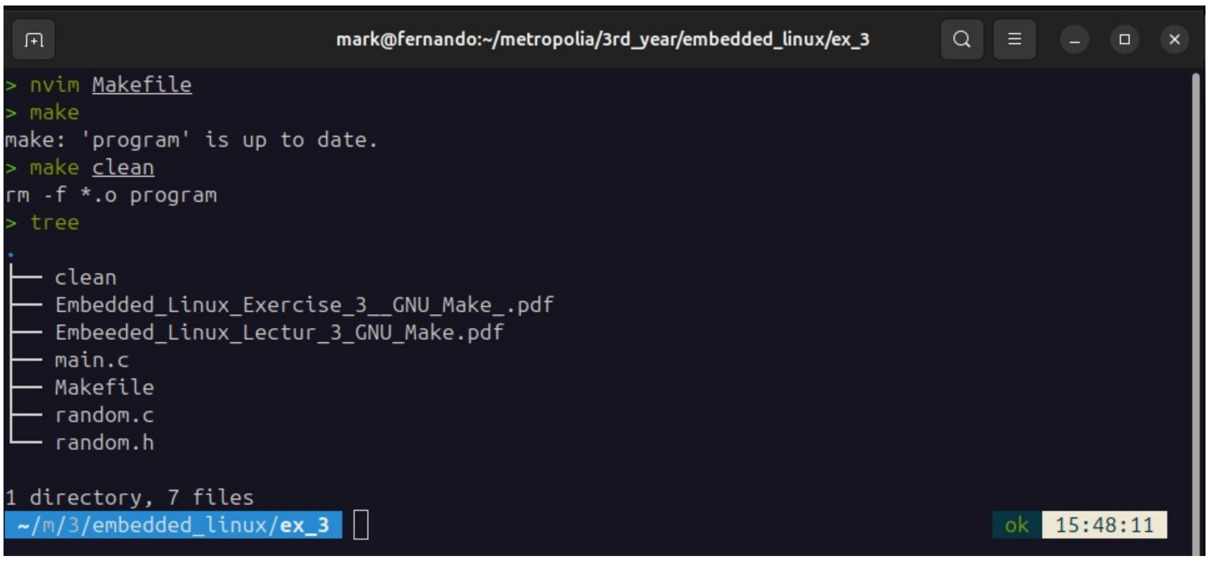


```
> touch clean
> make
gcc -c main.c -o main.o
gcc -c random.c -o random.o
gcc main.o random.o -o program
> tree
.
├── clean
├── Embedded_Linux_Exercise_3__GNU_Make_.pdf
├── Embeeded_Linux_Lectur_3_GNU_Make.pdf
├── main.c
├── main.o
├── Makefile
├── program
├── random.c
├── random.h
└── random.o

1 directory, 10 files
> make clean
make: 'clean' is up to date.
> tree
.
├── clean
├── Embedded_Linux_Exercise_3__GNU_Make_.pdf
├── Embeeded_Linux_Lectur_3_GNU_Make.pdf
├── main.c
├── main.o
├── Makefile
├── program
├── random.c
├── random.h
└── random.o

1 directory, 10 files
~/m/3/embedded_linux/ex_3                                    ok  15:43:57
```

It doesn't recognize the clean command, and it says clean is up to date since it did not change. Then I add clean with .PHONY. Then it worked.



```
mark@fernando:~/metropolia/3rd_year/embedded_linux/ex_3
> nvim Makefile
> make
make: 'program' is up to date.
> make clean
rm -f *.o program
> tree
.
├── clean
├── Embedded_Linux_Exercise_3__GNU_Make_.pdf
├── Embeeded_Linux_Lectur_3_GNU_Make.pdf
├── main.c
├── Makefile
├── random.c
└── random.h

1 directory, 7 files
~/m/3/embedded_linux/ex_3                                    ok  15:48:11
```

Compact Makefile

Next.

Src = $(wildcard *.c) -> Inlcuding all .c files in the current directory

Obj = $(src:.c = .o) -> This represents the object files. This command replaces all .c files with .o (main.c, random.c --> main.o, random.o). But here this not creating object files, just renaming them.

target = program --> 'program' is the name of final executable file

all: $(target) --> Setting the target to be 'program', which depends on $(target)

$(target): $(obj) --> To build $(target), we need all object files
$(g) $(obj) -o $(target) This links all the object files into the final executable 'program

%.o: %.c --> This is the place where actual .o files are
$(g) $(fl) -c $< -o $@ created. Here both < and @ are automatic variables. $< refers to the input .c files. $@ refers to the output .o file.

```
clean:
rm -f $(obj)                    --> Removing the object files


.PHONY: all clean               --> Declaring 'all' and 'clean' as phony, avoid
                                      conflicts with actual files
```