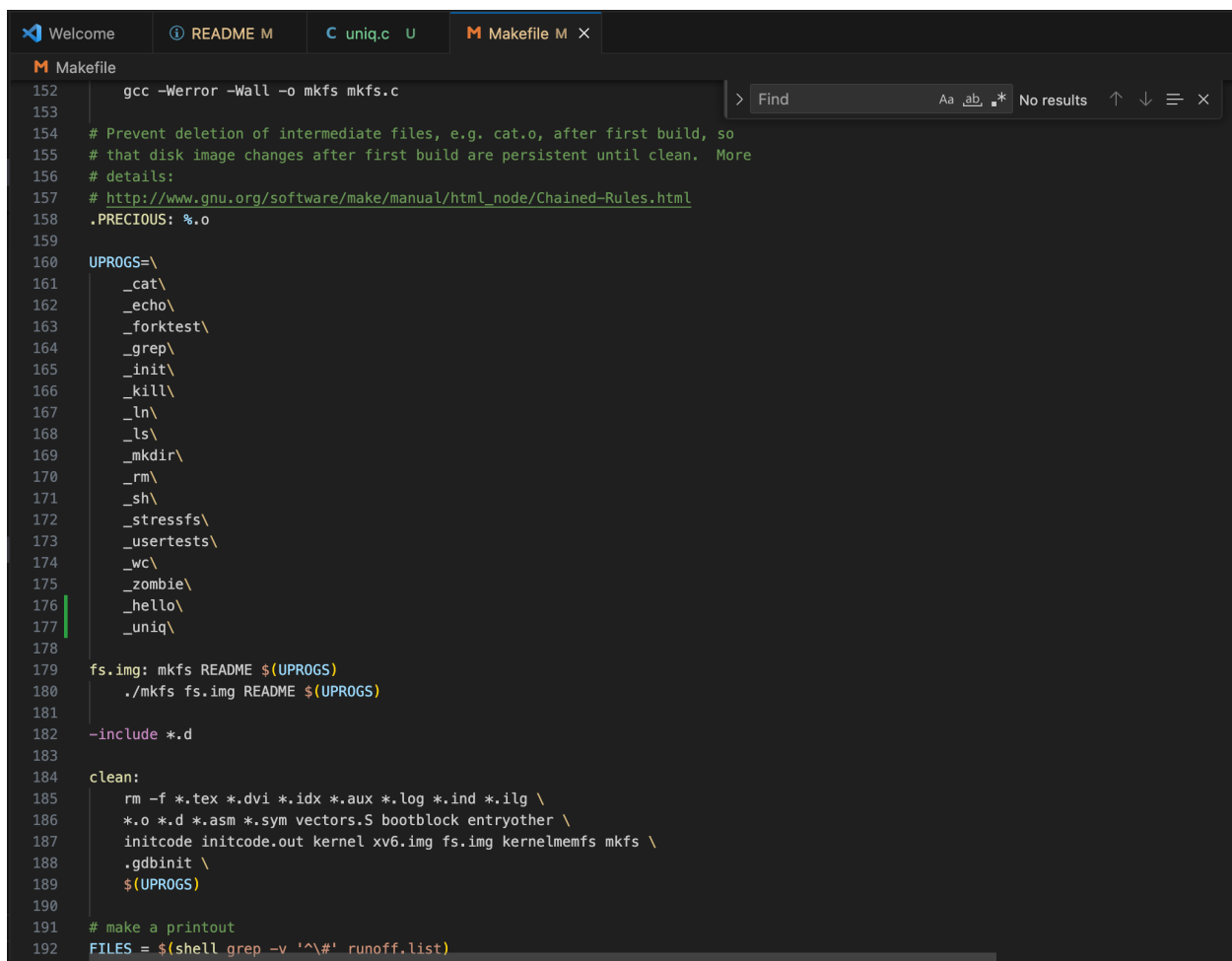


Hari Kishan Reddy Abbasani (ha2755)  
Bharani Kumar Reddy (bb3722)

## INSTRUCTIONS TO CHANGE IN MAKEFILE

- Add `_uniq\` under UPROGS as shown in Makefile:



```
152 gcc -Werror -Wall -o mkfs mkfs.c
153
154 # Prevent deletion of intermediate files, e.g. cat.o, after first build, so
155 # that disk image changes after first build are persistent until clean. More
156 # details:
157 # http://www.gnu.org/software/make/manual/html\_node/Chained-Rules.html
158 .PRECIOUS: %.o
159
160 UPROGS=\
161     _cat\
162     _echo\
163     _forktest\
164     _grep\
165     _init\
166     _kill\
167     _ln\
168     _ls\
169     _mkdir\
170     _rm\
171     _sh\
172     _stressfs\
173     _usertests\
174     _wc\
175     _zombie\
176     _hello\
177     _uniq\
178
179 fs.img: mkfs README $(UPROGS)
180     ./mkfs fs.img README $(UPROGS)
181
182 -include *.d
183
184 clean:
185     rm -f *.tex *.dvi *.idx *.aux *.log *.ind *.ilg \
186         *.o *.d *.asm *.sym vectors.S bootblock entryother \
187         initcode initcode.out kernel xv6.img fs.img kernelmemfs mkfs \
188         .gdbinit \
189         $(UPROGS)
190
191 # make a printout
192 FILES = $(shell grep -v '^#' runoff.list)
```

- Add -std=c99 tag

Add the following line as shown in the below figure.

CFLAGS = -std=c99 -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector

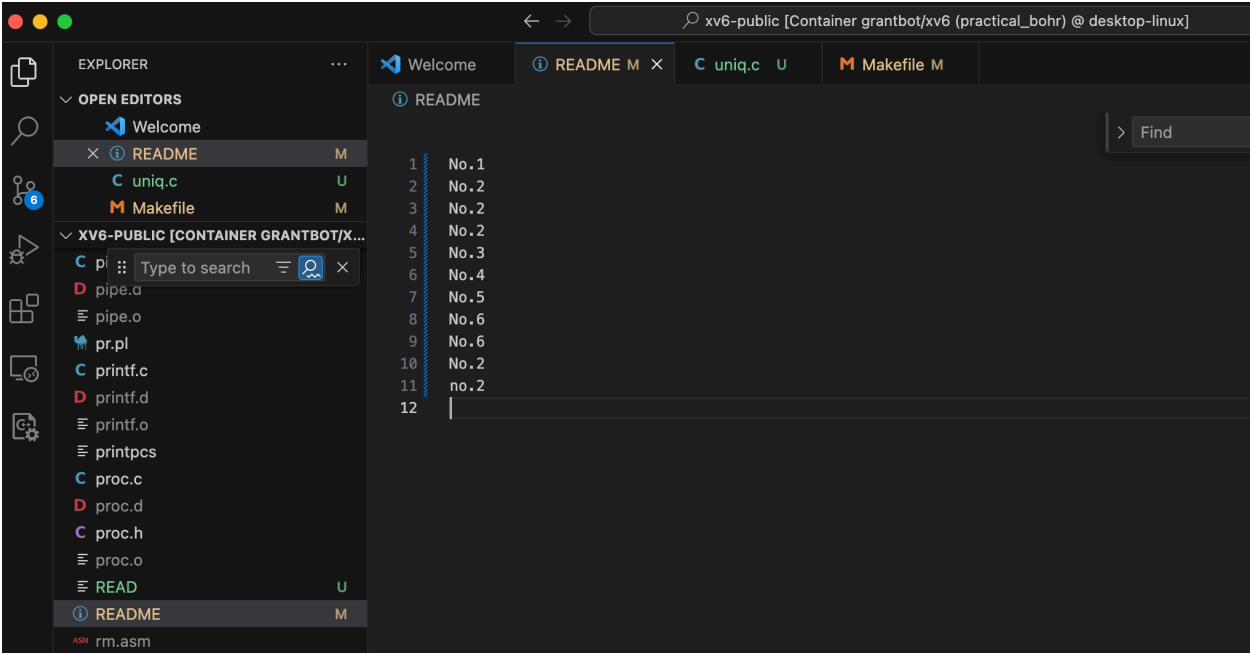
The screenshot shows a VS Code editor window with a project named 'xv6-public'. The Explorer sidebar on the left shows the file structure, including 'Makefile', 'log.c', 'log.d', 'log.o', 'ls.asm', 'ls.c', 'ls.d', 'ls.o', 'ls.sym', 'main.c', 'main.d', 'main.o', 'memide.c', 'memlayout.h', 'mkdir.asm', 'mkdir.c', 'mkdir.d', 'mkdir.o', 'mkdir.sym', 'mkfs', 'mkfs.c', 'mmu.h', 'mp.c', 'mp.d', 'mp.h', and 'mp.o'. The main editor area displays the 'Makefile' with the following content:

```
73 CC = $(TOOLPREFIX)gcc
74 AS = $(TOOLPREFIX)gas
75 LD = $(TOOLPREFIX)ld
76 OBJCOPY = $(TOOLPREFIX)objcopy
77 OBJDUMP = $(TOOLPREFIX)objdump
78 CFLAGS = -std=c99 -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector
79 #CFLAGS = -fno-pic -static -fno-builtin -fno-strict-aliasing -fvar-tracking -fvar-tracking-assignments -O0 -g -Wall -MD -gdwarf-2 -m32
80 CFLAGS += $(shell $(CC) -fno-stack-protector -E -x c /dev/null >/dev/null 2>&1 && echo -fno-stack-protector)
81 ASFLAGS = -m32 -gdwarf-2 -Wa,-divide
82 # FreeBSD ld wants ``elf_i386_fbsd''
83 LDFLAGS += -m $(shell $(LD) -V | grep elf_i386 2>/dev/null | head -n 1)
84
```

The terminal at the bottom shows the output of the 'make' command. It indicates that the Makefile is being executed, and the compiler is using the specified flags. The output shows the compilation of 'main.c' into 'main.o' and the linking of the final executable 'xv6'.

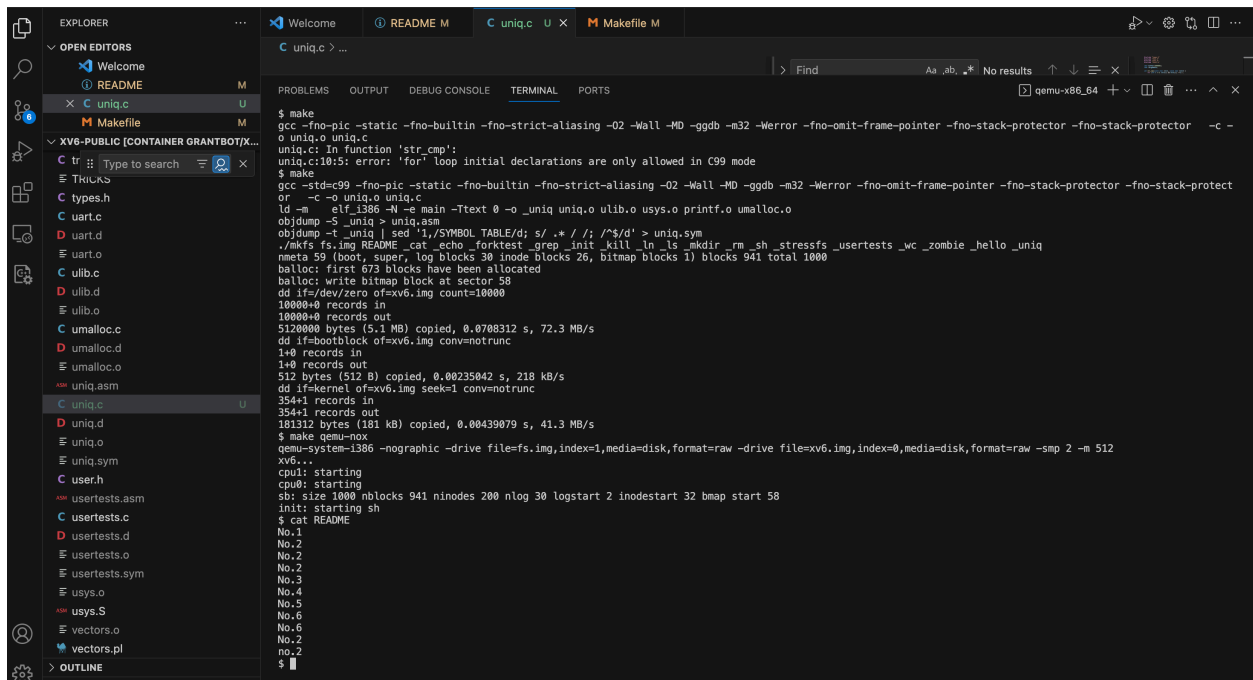
```
$ make
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-stack-protector -c -o main.o main.c
ld -m elf_i386 -N -e main -Ttext 0 -o _uniq _uniq.o _lib.o _sys.o _printf.o _umalloc.o
objdump -t _uniq | sed '1,/SYMBOL TABLE/d; s/ ./ /; /$/d' > _uniq.sym
./mkfs fs.img README_cat_echo_forktest_grep_init_kill_ln_ls_mkdir_rm_sh_stressfs_userstests_wc_zombie_hello_uniq
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 941 total 1000
balloc: first 673 blocks have been allocated
balloc: write bitmap block at sector 58
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB) copied, 0.0683821 s, 74.9 MB/s
dd if=bootblock of=xv6.img conv=notrunc
140 records in
140 records out
512 bytes (512 B) copied, 0.00224079 s, 228 kB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
354+1 records in
354+1 records out
181312 bytes (181 kB) copied, 0.00584213 s, 31.0 MB/s
$
```

README FILE



## SAMPLE OUTPUTS

- cat README



The screenshot shows a VS Code editor interface with a terminal window open. The terminal displays the output of a Makefile build and execution. The build process uses GCC with various flags to compile a C program into an ELF executable. The execution output shows the program's behavior, including file operations and system calls.

```
$ make
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-stack-protector -c -o uniq.o uniq.c
uniq.c: In function 'str_cmp':
uniq.c:10:5: error: 'for' loop initial declarations are only allowed in C99 mode
$ make
gcc -std=c99 -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-stack-protector -c -o uniq.o uniq.c
ld -m elf_i386 -N -e main -Ttext 0 -o _uniq uniq.o ulib.o usys.o printf.o umalloc.o
objdump -S _uniq > uniq.asm
objdump -t _uniq | sed '1,/SYMBOL TABLE/d; s/.* //; /"/d' > uniq.sym
./mkfs.fs.img README_cat_echo_forktest_grep_init_kill_ln_ls_mkdir_rm_sh_stressfs_usertests_wc_zombie_hello_uniq
mmta 59 (boot, super, log blocks 30 inode blocks 26, Bitmap blocks 1) blocks 941 total 1000
ballocc: first 673 blocks have been allocated
ballocc: write bitmap block at sector 58
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
512000 bytes (512 kB) copied, 0.0708312 s, 72.3 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.00235042 s, 218 kB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
354+1 records in
354+1 records out
181312 bytes (181 kB) copied, 0.00439079 s, 41.3 MB/s
$ make qemu-nox
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu0: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ cat README
No.1
No.2
No.2
No.2
No.3
No.4
No.5
No.6
No.6
No.2
no.2
$
```

- uniq README, Cat README | uniq

The screenshot shows a VS Code editor interface with a terminal window. The terminal output is as follows:

```

10000+0 records in
10000+0 records out
512000 bytes (5.1 MB) copied, 0.0708312 s, 72.3 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.00235042 s, 218 kB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
354+1 records in
354+1 records out
181312 bytes (181 kB) copied, 0.00439079 s, 41.3 MB/s
$ make qemu-nox
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ cat README
No.1
No.2
No.2
No.3
No.4
No.5
No.6
No.6
No.2
no.2
$ uniq README
No.1
No.2
No.3
No.4
No.5
No.6
No.2
no.2
$ cat README | uniq
No.1
No.2
No.3
No.4
No.5
No.6
No.2
no.2
$

```

The Explorer sidebar on the left shows the project structure, including files like 'README', 'Makefile', and various source files such as 'types.h', 'uart.c', 'umalloc.c', 'umalloc.d', 'umalloc.o', 'uniq.asm', 'uniq.c', 'uniq.d', 'uniq.o', 'uniq.sym', 'user.h', 'usertests.asm', 'usertests.c', 'usertests.d', 'usertests.o', 'usertests.sym', 'usys.o', 'usys.S', 'vectors.o', and 'vectors.pl'.

- `uniq -c README`
- `uniq -d README`
- `uniq -i README`
- `uniq -c -i README`

The screenshot shows a VS Code editor window with the following components:

- EXPLORER:** A file tree on the left showing the project structure. The file `uniq.c` is selected.
- EDITOR:** The main workspace showing the source code of `uniq.c`. The code includes headers for `types.h`, `stat.h`, `user.h`, and `fcntl.h`. It defines a `buffer` of size 1000000 and a function `str_cmp` that compares two strings character by character, converting them to lowercase.
- TERMINAL:** A terminal window at the bottom showing the execution of the `uniq` command on the `README` file. The output shows the results of the command, including the number of occurrences of each line and the line numbers.

```

1 #include "types.h"
2 #include "stat.h"
3 #include "user.h"
4 #include "fcntl.h"
5
6 char buffer[1000000];
7 char *arguments;
8
9 int str_cmp(const char *str1, const char *str2) {
10     for (int i = 0; str1[i] && str2[i]; ++i) {
11         char char1 = str1[i] | 32; // Convert to lowercase
12         char char2 = str2[i] | 32; // Convert to lowercase
13         if (char1 != char2) {
14             return char1 - char2;
15         }
16     }
17     return 0;
18 }

```

```

$ uniq -c README
1 No.1
3 No.2
1 No.3
1 No.4
1 No.5
2 No.6
1 No.2
1 no.2
$ uniq -d README
No.2
No.6
$ uniq -i README
No.1
No.2
No.3
No.4
No.5
No.6
No.2
$ uniq -c -i README
1 No.1
3 No.2
1 No.3
1 No.4
1 No.5
2 No.6
2 No.2
$

```