# Project 1: Implementing shell commands in xv6

## Part 1 Task 1 - Implementing the `uniq` command in user mode

**System Environment**

Operating System: XV6

Compiler/Development Environment: GCC (GNU Compiler Collection).

Language Used: C

**Overview**

This code implements a custom uniq utility in C, which reads input from either files or stdin and removes duplicate lines while providing various options for customization. The main functionalities include:

Removing duplicate lines.

Counting duplicate lines with the -c flag.

Displaying only duplicate lines with the -d flag.

Enabling case-insensitive comparison with the -i flag.

**Code Structure**

**`strcasecmp` Function**

This custom function enables case-insensitive string comparison.

It employs bitwise operations to compare characters while ignoring case differences.

**`custom\_uniq` Function**

The core function for processing input and removing duplicates.

Utilizes arrays like pilot, output, and repeat to store line data, unique lines, and their counts.

Reads data from a file descriptor (or stdin) into buf, processing input line by line.

Stores unique lines and counts, then prints results based on command-line arguments.

Frees dynamically allocated memory.

**main Function**

Entry point of the program.

Initializes variables, including file\_descriptor and case\_insensitive.

Handles different scenarios based on the number and values of command-line arguments.

Supports case-insensitive search with the -i flag.

**Approach**

* Header Inclusion: I started by including the necessary header files (`types.h`, `stat.h`, `user.h`) to ensure compatibility with file operations and standard I/O.
* Case-Insensitive Comparison: I implemented the `strcasecmp` function using bitwise operations to enable case-insensitive string comparison. This function allows for accurate comparisons while ignoring case differences.
* `custom\_uniq` Function: This function serves as the core of the utility. It reads input from a file descriptor (or stdin) and processes it to remove duplicate lines. I used arrays like `pilot`, `output`, and `repeat` to store data and counts. The processing occurs line by line, with unique lines and their counts stored in the `output` and `repeat` arrays. The function prints the results based on command-line arguments and frees dynamically allocated memory.
* `main` Function: As the entry point of the program, the main initializes essential variables such as `file\_descriptor` and `case\_insensitive`. It handles various scenarios based on the number and values of command-line arguments (argc). Depending on the arguments provided, it enables case-insensitive search with the `-i` flag.

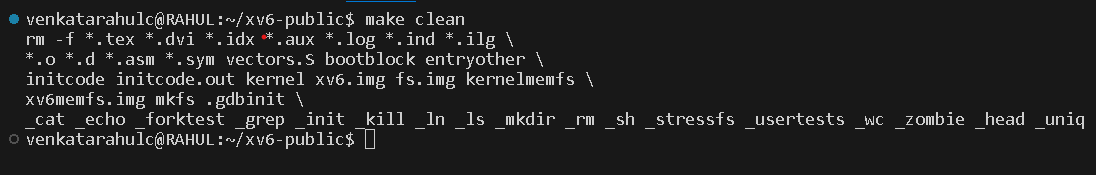
**Output Options**

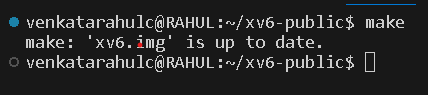
The code provides different output options based on command-line arguments:

* -c: Count duplicate lines.
* -d: Display only duplicate lines.
* -i: Enabling case-insensitive comparison.
* Default: Remove duplicate lines and display unique lines.

**Steps to Run**

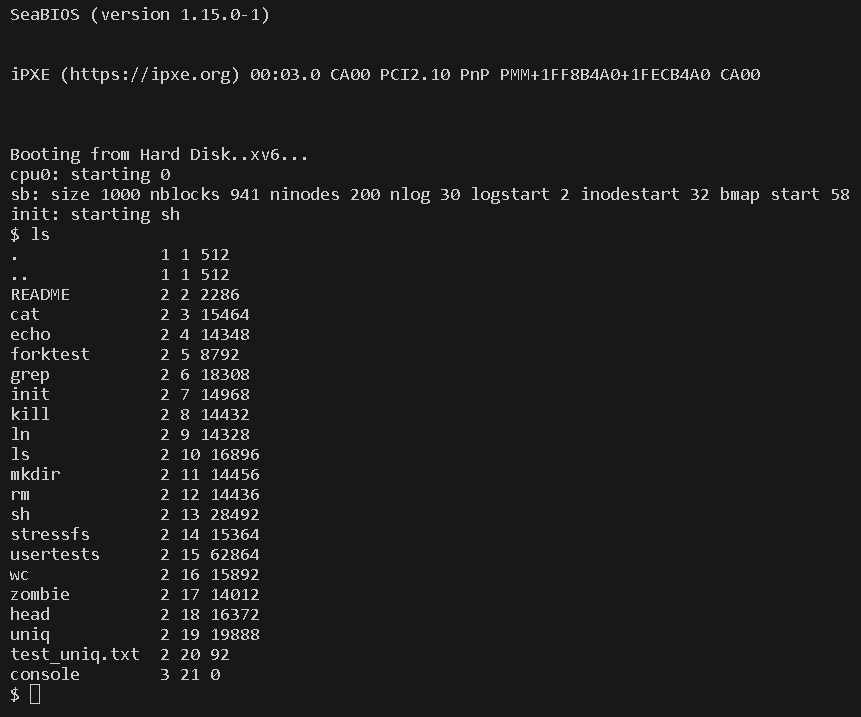
* **Compile the Program**
  + Using the Xv6 build system to compile the custom uniq utility.



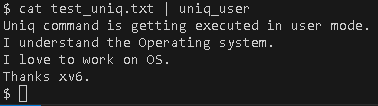


* **Access the Xv6 Environment**
  + Boot or launch the Xv6 operating system on the system or in an emulator, such as QEMU

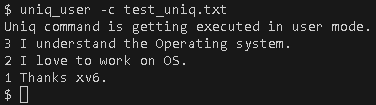




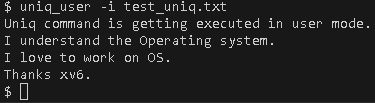
* **Run the Program**
  + cat “filename” | uniq\_user



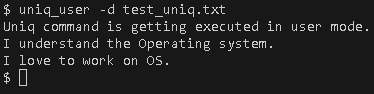
* + uniq\_user -c “filename”



* + uniq\_user -i “filename”



* + uniq\_user -d “filename”



**Resources Used**

* [**https://viduniwickramarachchi.medium.com/how-to-add-a-user-program-to-xv6-1209069feee4**](https://viduniwickramarachchi.medium.com/how-to-add-a-user-program-to-xv6-1209069feee4)
* [**https://www.youtube.com/watch?v=fWUJKH0RNFE&list=PLbtzT1TYeoMhTPzyTZboW\_j7TPAnjv9XB&pp=iAQB**](https://www.youtube.com/watch?v=fWUJKH0RNFE&list=PLbtzT1TYeoMhTPzyTZboW_j7TPAnjv9XB&pp=iAQB)
* **XV6 Documentation**
* **Group Effort (Mukesh Padireti, Anthony McCofie)**

## Part 1 Task 2 - Implementing the `uniq` command in kernel mode

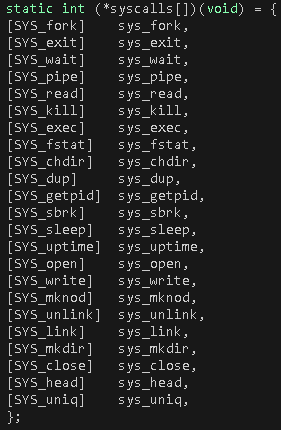
**Procedure**

The following files have been modified for adding the `uniq` system call in XV6:

* **syscall.c**

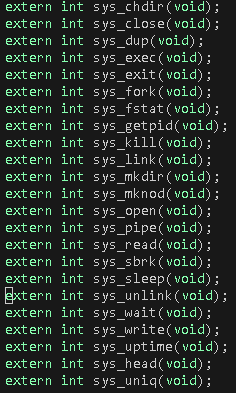
adding below entry in this array

**[SYS\_uniq] sys\_uniq,**



Also adding the function prototype to this file

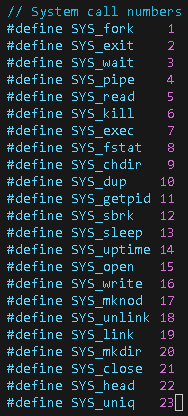
**extern int sys\_uniq(void);**



* **syscall.h**

There is an array of function pointers in the file syscall.c to index in this array, we will define the number of a system call in syscall.h file. This number is used for indexing in that array of function pointers.

**#define SYS\_uniq 23**

****

* **sysproc.c**

Implemented the system call in this file.

**`strcasecmp` Function**

This custom function enables case-insensitive string comparison.

It employs bitwise operations to compare characters while ignoring case differences.

**`sys\_uniq` Function**

The core function for processing input and removing duplicates.

Utilizes arrays like pilot, output, and repeat to store line data, unique lines, and their counts.

Reads data from a file descriptor (or stdin) into buf, processing input line by line.

Stores unique lines and counts, then prints results based on command-line arguments.

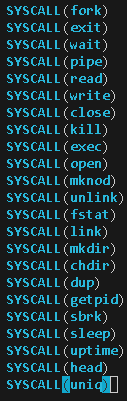
Frees dynamically allocated memory.

Returns an integer data specifying the status of function execution.

* **usys.S**

The interface is added in usys.S file.

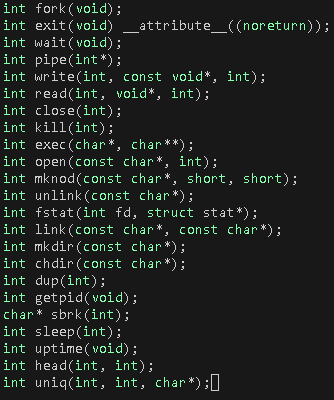
**SYSCALL (uniq)**

****

* **user.h**

Function Prototype which the user program will call is added in user.h file.

Int uniq(int, int, char\*);



* **User Program named uniq.c**

Entry point of the program.

Initializes variables, including file\_descriptor and case\_insensitive.

Handles different scenarios based on the number and values of command-line arguments.

Supports case-insensitive search with the -i flag.

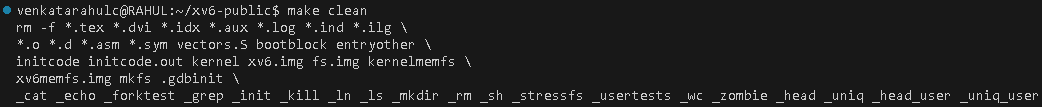
**Output Options**

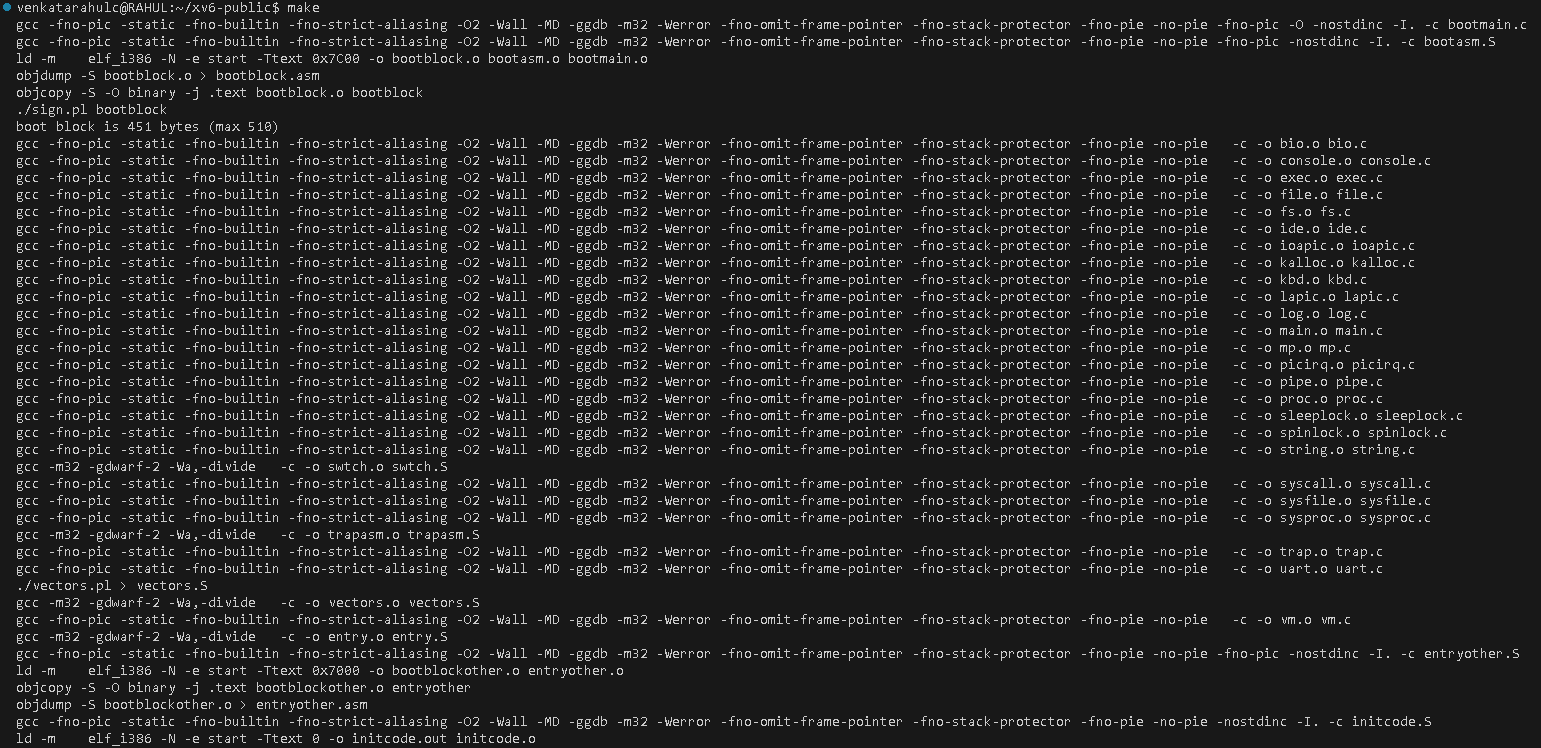
The code provides different output options based on command-line arguments:

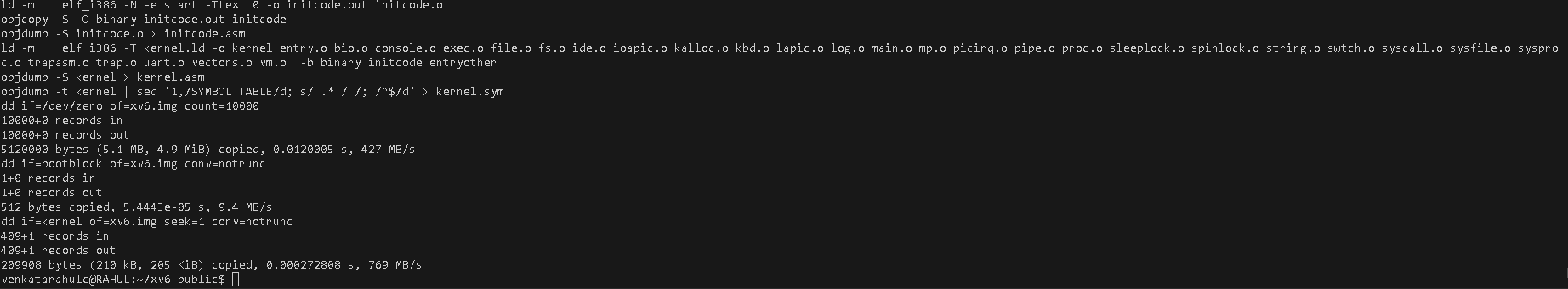
* -c: Count duplicate lines.
* -d: Display only duplicate lines.
* -i: Enabling case-insensitive comparison.
* Default: Remove duplicate lines and display unique lines.

**Steps to Run**

* **Compile Your Program**
  + Use the Xv6 build system to compile the custom uniq utility.

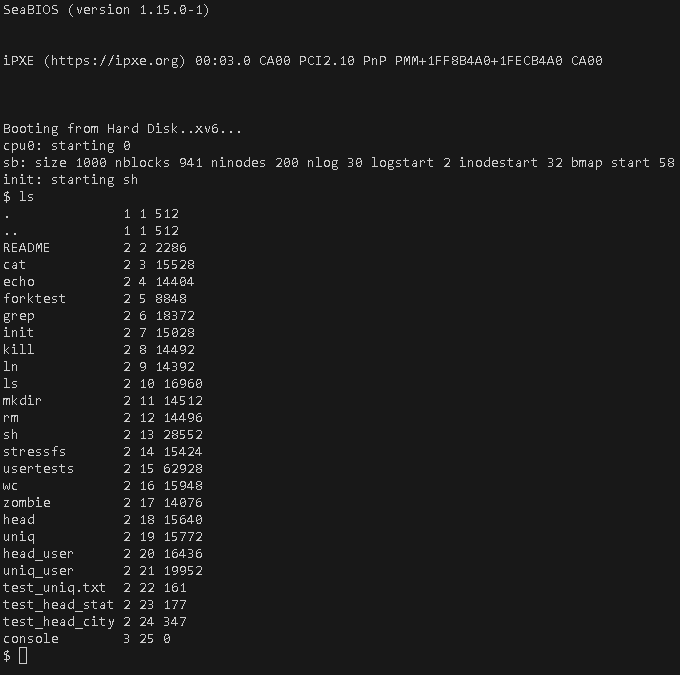




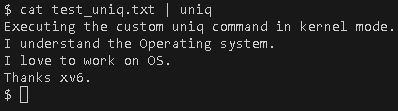


* **Access the Xv6 Environment**
  + Boot or launch the Xv6 operating system on the system or in an emulator, such as QEMU

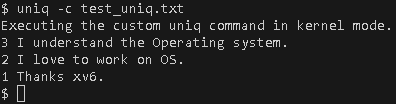




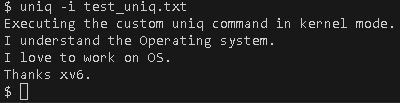
* **Run the Program**
  + cat “filename” | uniq



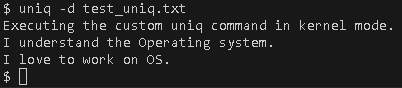
* + uniq -c “filename”



* + uniq -i “filename”



* + uniq -d “filename”



**Resources Used**

* [**https://medium.com/@mahi12/adding-system-call-in-xv6-a5468ce1b463**](https://medium.com/@mahi12/adding-system-call-in-xv6-a5468ce1b463)
* [**https://www.youtube.com/watch?v=fWUJKH0RNFE&list=PLbtzT1TYeoMhTPzyTZboW\_j7TPAnjv9XB&pp=iAQB**](https://www.youtube.com/watch?v=fWUJKH0RNFE&list=PLbtzT1TYeoMhTPzyTZboW_j7TPAnjv9XB&pp=iAQB)
* **XV6 Documentation**
* **Group Effort (Mukesh Padireti, Anthony McCofie)**