# Project 1: Implementing shell commands in xv6

## Part 2 Task 1 - Implementing the `head` command in user mode

**System Environment**

Operating System: XV6

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

Language Used: C

**Overview**

The head command is primarily used for displaying the initial portion of text files or input data, making it useful for tasks such as:

Previewing the content of files quickly.

Checking the header or introductory lines of data files.

Extracting a specific number of lines or bytes from the beginning of a file.

**Code Structure**

**`custom\_head` Function**

`custom\_head` reads and prints the first `num\_lines` from a given `file\_descriptor`.

It maintains a line buffer to collect characters of each line.

The function tracks the line number and terminates when the desired number of lines has been printed.

**main Function**

`main` is the entry point of the program.

It initializes variables and the default number of lines to print.

The code processes command-line arguments to customize the number of lines and filenames to display.

It invokes the `custom\_head` function to print the specified lines or the default lines from stdin.

**Approach**

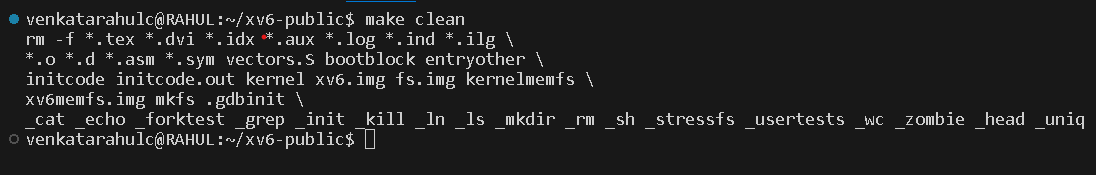
* Header Inclusion: I have used `#include` directives to include essential header files like `types.h`, `stat.h`, and `user.h`. These headers provide necessary functionalities for file operations and system calls.
* Custom Head Function (`custom\_head`): Within the code, I have implemented a custom function named `custom\_head`. This function serves as the core of the "head" command utility and is responsible for processing input and displaying the specified number of lines from the beginning of a file or standard input.
* Buffer for Data Storage: To efficiently read and process data, I have defined a buffer named `buffer` with a size of 512 bytes. This buffer is used for reading data from files or standard input.
* Line and Character Counting: Inside the `custom\_head` function, I have employed variables like `line\_number` and `char\_count` to keep track of the current line number and character count within the input.
* Line Buffer (`line\_buffer`): To temporarily store lines that need to be displayed, I have used a character array called `line\_buffer`. This array accumulates characters until a complete line is ready for output.
* Reading and Displaying Data: The code reads data in chunks from the input and processes it character by character. It checks for newline characters to identify the end of lines. When the specified number of lines (`num\_lines`) is reached, the function returns, effectively stopping further processing. It prints lines as they are read and formatted with line numbers.
* Main Function: The main function serves as the entry point of the program. It handles command-line arguments and orchestrates the execution of the "head" command.

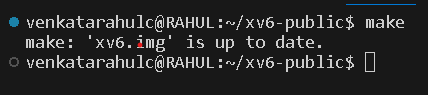
**Output Options**

It checks for the `-n` option and validates the following argument to set the number of lines accordingly.

**Steps to Run**

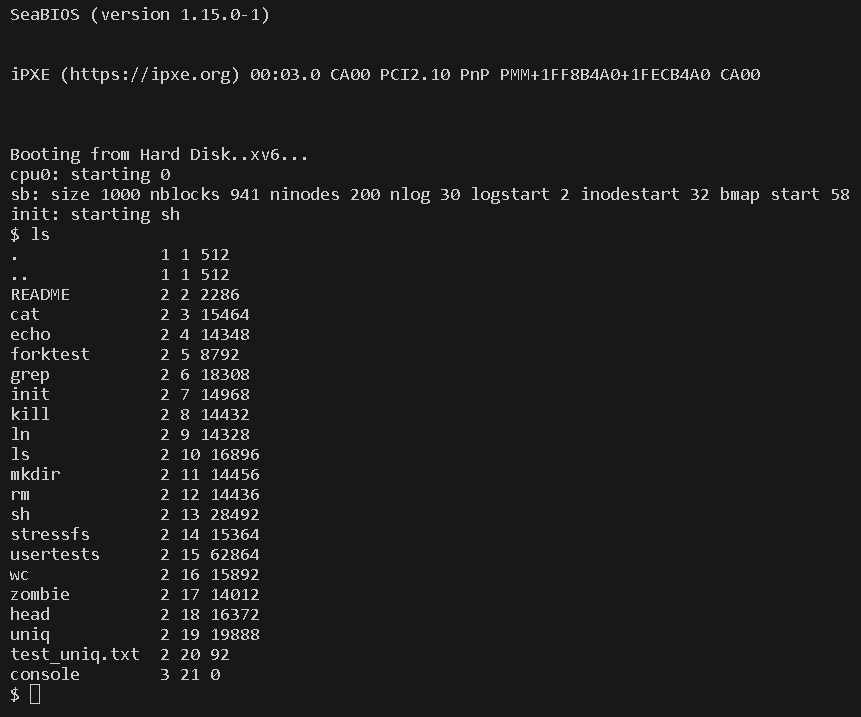
* **Compile the Program**
  + Use the Xv6 build system to compile the custom head utility.



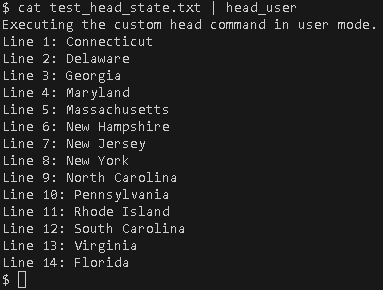


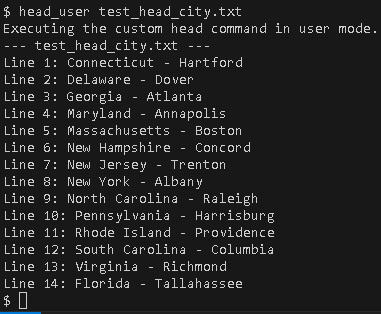
* **Access the Xv6 Environment**
  + Boot or launch the Xv6 operating system on the system or in an emulator, such as QEMU



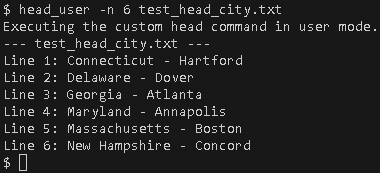


* **Run the Program**
  + head\_user “filename” (or) cat “filename” | head\_user 🡪 default 14 lines

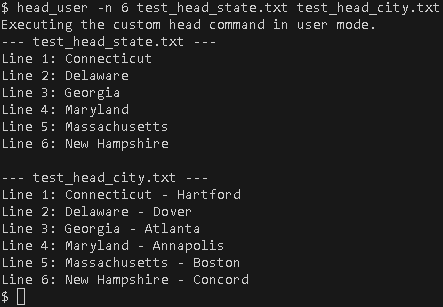




* + head\_user – n “number” “filename”



* + head\_user – n “number” “filename1” ”filename2”



**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 `head` command in kernel mode

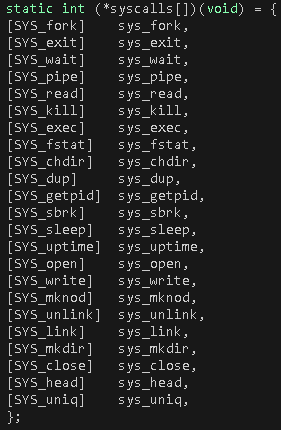
**Procedure**

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

* **syscall.c**

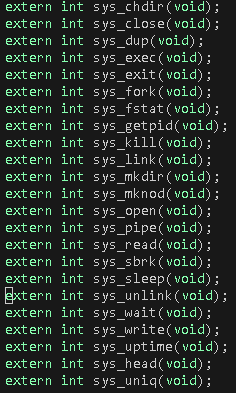
adding below entry in this array

**[SYS\_head] sys\_head,**



Also adding the function prototype to this file

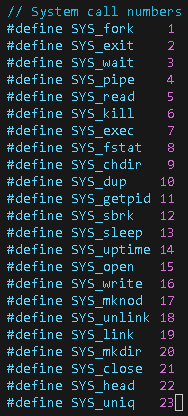
**extern int sys\_head(void);**



* **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\_head 22**

****

* **sysproc.c**

Implemented the system call in this file.

**`sys\_head` Function**

`custom\_head` reads and prints the first `num\_lines` from a given `file\_descriptor`.

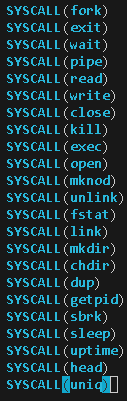
It maintains a line buffer to collect characters of each line.

The function tracks the line number and terminates when the desired number of lines has been printed.

* **usys.S**

The interface is added in usys.S file.

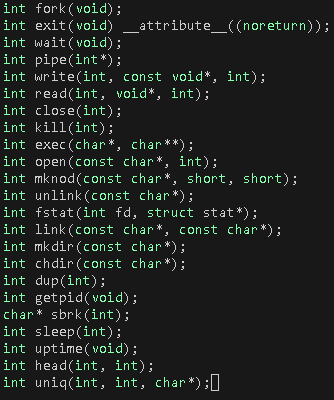
**SYSCALL (head)**

****

* **user.h**

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

int head (int, int);



* **User Program named head.c**

main Function

`main` is the entry point of the program.

It initializes variables and the default number of lines to print.

The code processes command-line arguments to customize the number of lines and filenames to display.

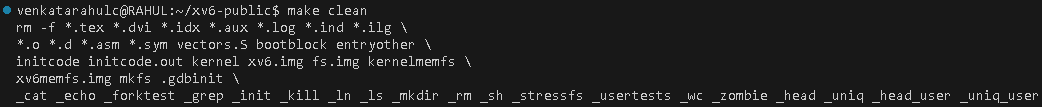
It invokes the `custom\_head` function to print the specified lines or the default lines from stdin.

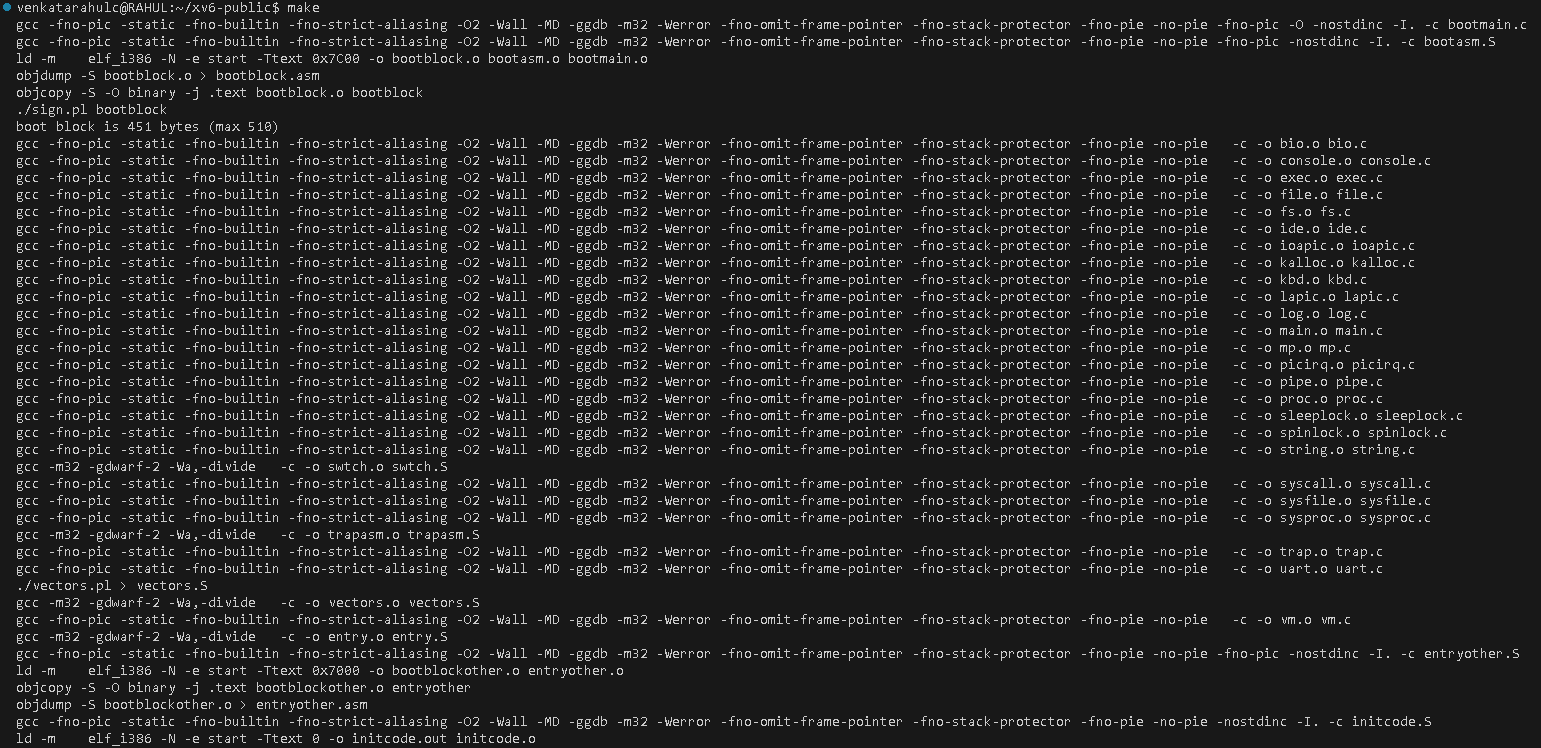
**Output Options**

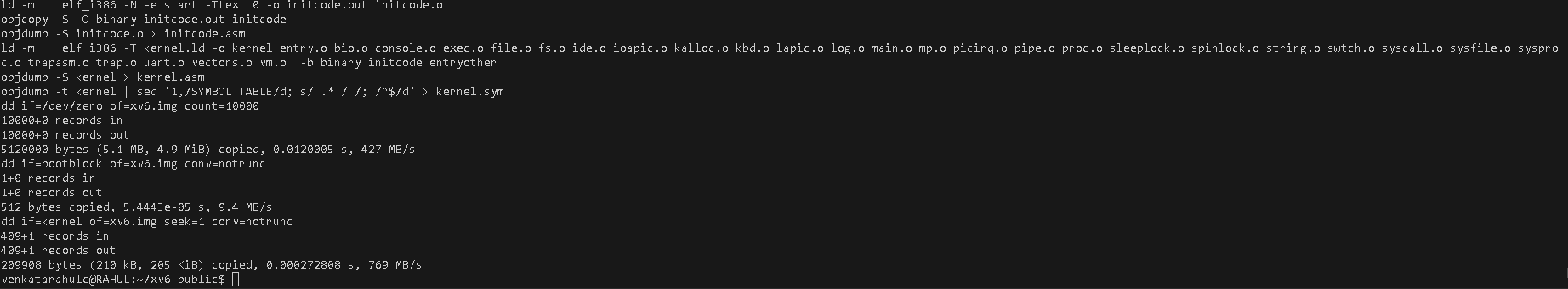
It checks for the `-n` option and validates the following argument to set the number of lines accordingly.

**Steps to Run**

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

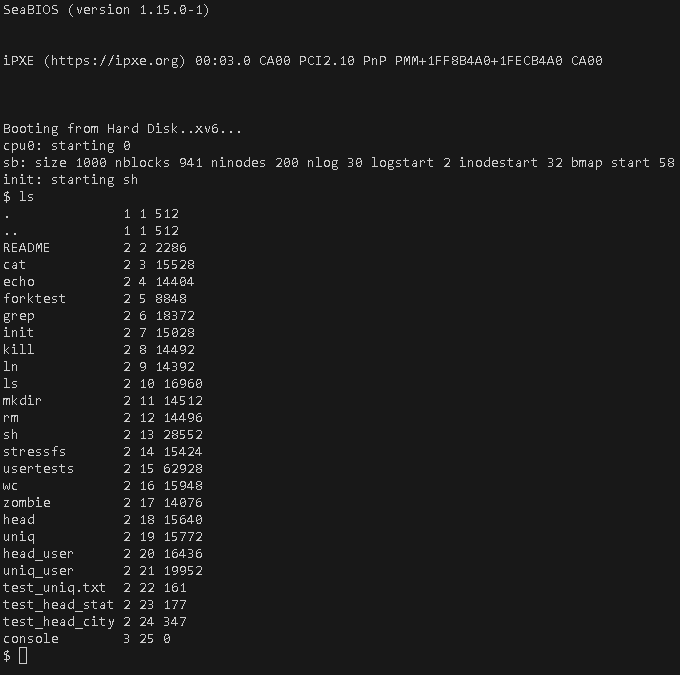




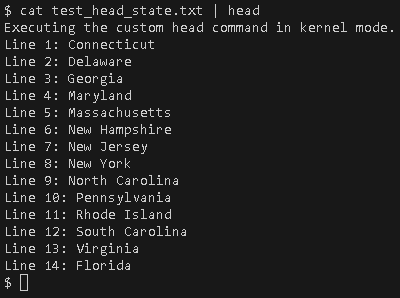


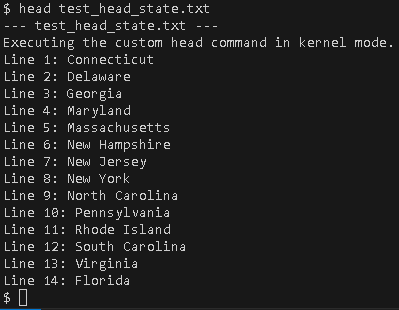
* **Access the Xv6 Environment**
  + Boot or launch the Xv6 operating system on the system or in an emulator, such as QEMU



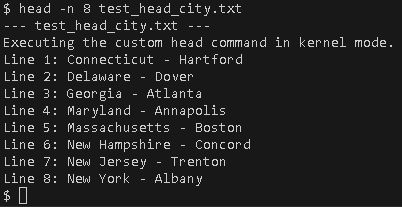


* **Run the Program**
  + head “filename” (or) cat “filename” | head 🡪 default 14 lines

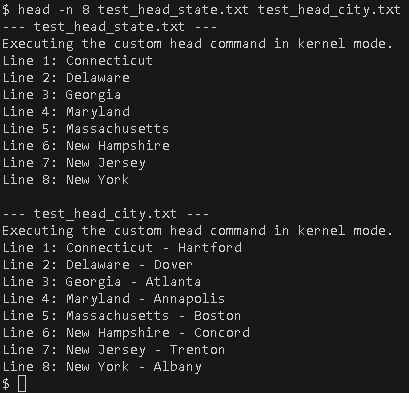




* + head – n “number” “filename”



* + head – n “number” “filename1” ”filename2”



**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)**