Assignment 2A

Group 26

Team Members

Name	**Roll No.**
Abhishek Agrahari	190123066
Manish Kumar	190123067
Vivek Kumar	190101100
Anubhav Bajaj	190123007
Uday Singh	190101095

Task 1.)

a)

- defs.h : added function prototype “int history(char*, int);” at line 24.
- sh.c : it checks if the first 7 characters of the command are ‘history’ then it makes system calls to print all the stored commands (maximum 16) used in the past.
- console.c :

  Defined arrows to their ASCII values and MAX_HISTORY as 16.

  Made a struct ‘records’ for storing the commands executed in the past. A maximum of 16 commands could be stored.

  We implemented the following functions:

1) copy_buffer() - Copy input.buf to a safe location. Used only when punching in new keys and the caret isn't at the end of the line.
2) shift_buffer_left() - it shifts the characters on the right of the caret to the left on pressing backspace when the caret is not at the end.
3) shift_buffer_right() - it shifts the characters on the right of the caret to the right on pressing new keys when the caret is not at the end.
4) history(char *buffer, int historyId) - this is the function that gets called by 'sys_history' in sysproc.c and writes the requested command history in the buffer.
5) consoleintr() - it specifies the work on pressing any of the arrows.

   Otherwise it updates the commands in the command history table.

b)

All the below files have been attached in the folder. The following files were edited to add the system call:

- syscall.h : This file assigns a number to every system call in xv6 system. By default, xv6 has initially 21 system calls. I added "SYS_history" and assigned it number 22.

  Line 23 : #define SYS_history 22

- user.h : The function prototype is added in user.h.

  Line 26: int history(char*, int);

- usys.S : The code to add it as a system call is included in usys.S

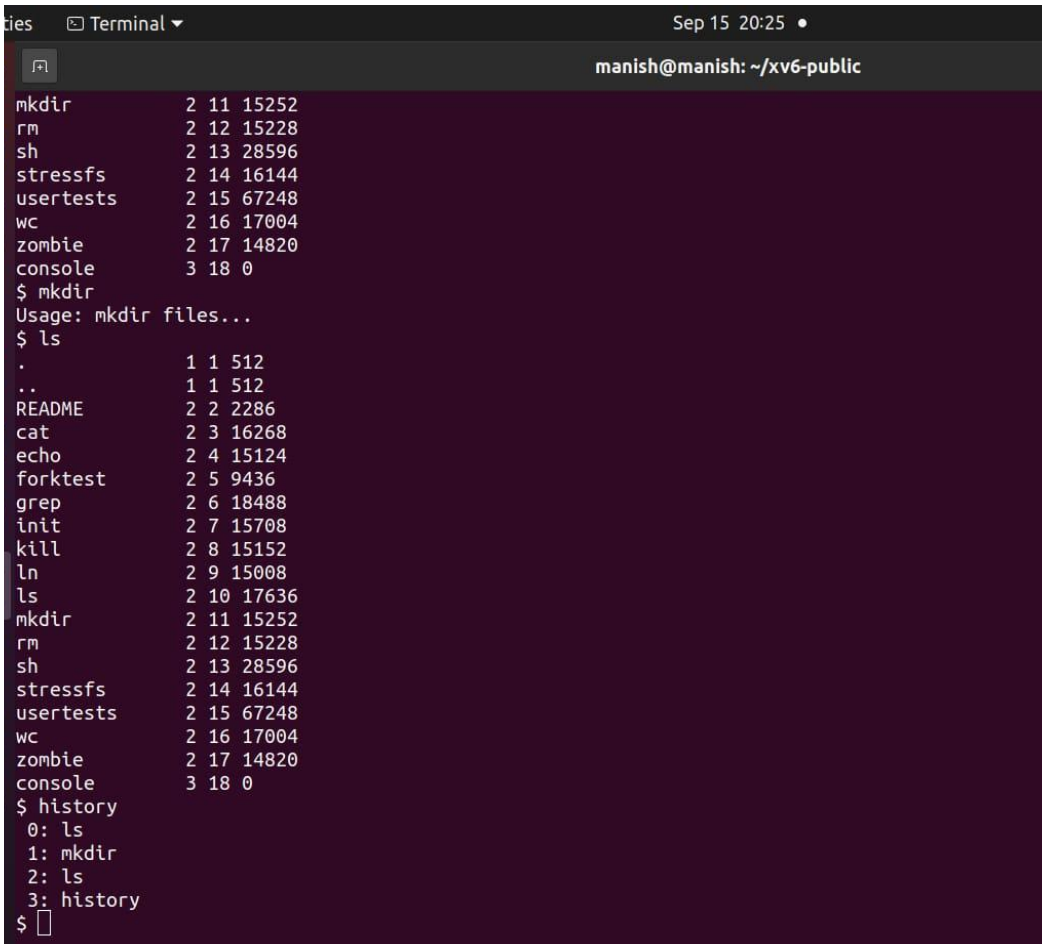
  Line 32: SYSCALL(history)

- syscall.c : The function prototype is put here using extern keyword. It also contains an array of function pointers(syscalls[]) which uses index defined in syscall.h. So I added sys_history at index 22 in syscalls[].

  Line106 : extern int sys_history(void);

  Line130 : [SYS_history] sys_history,

- sysproc.c : This file contains the defination of sys_history.



Task 2.)

For implementation of task two we did following things:

1. In proc.h file, added ctime, stime, retime, rutime fields in proc struct.

2. When the process is first created, updated ctime to the current clock tick and initialised stime, rutime and retime to 0.

3. The states of processes are already defined in proc struct. So in trap.c where clock tick is handled, we checked for the state of process and then according to the state of the process updated stime , retime or rutime of the process at every clock tick .

4. After this we implemented int wait2(int* retime, int* runtime, int* stime) system call which calls wait2() function and returns pid of child process on success else returns 1. For implementing the system call we did required changes in syscall.h, syscall.c, sysproc.c, user.h, usys.S and defs.h.

5. wait2() system call invokes wait2() function which is an extension of wait() function and assigns retime, rutime, stime field values of the process present in kernel to the provided(pointed) variables.

6. Now we added a simple test program, Timetest.c, which utilises the wait2() system call and prints retime, rutime, stime and pid of child process on successful call and 1 otherwise.

7. In order to include the test file, we made changes at appropriate places in Makefile like in UPROGS added _Timetest and added Timetest.c in EXTRAS.

8. Console output shown below:

viv2002@DESKTOP-E4H57JO: /mnt/e/Documents/xv6

```
SeaBIOS (version 1.13.0-1ubuntu1.1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ Timetest
exec: fail
Runnable time = 0
 Running time = 0
 Sleeping time = 0
 return status = 1
Runnable time = 0
 Running time = 2
 Sleeping time = 0
 return status = 4
$
```