LAB – 2A Group No - 44

1.1

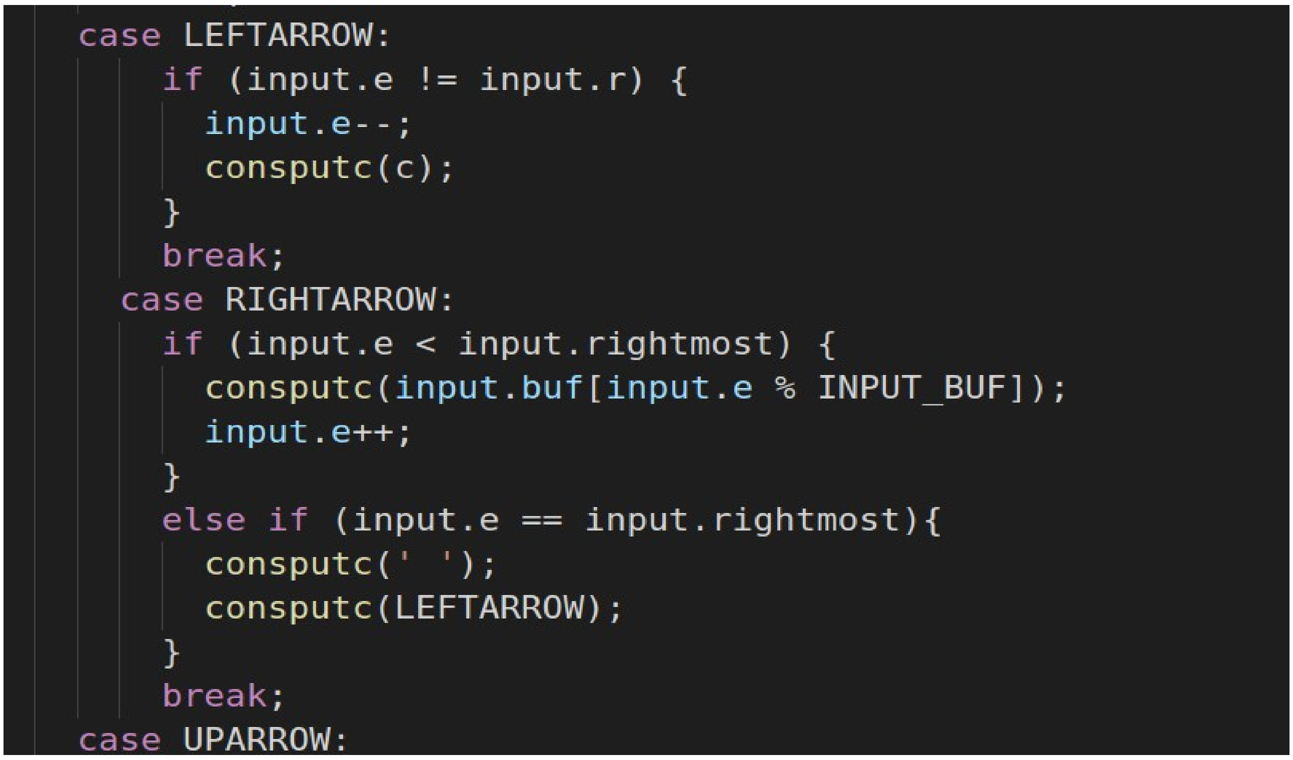
Adding Caret Navigation to the console :

when we first press a key on the keyboard the consoleintr raises an interpput for us and after the interpput througth the serial port the charecter is written to the console (which is a file in xv6) using cgaputc function. This what happpens when you want to input to the console.

To implement caret navigation we have to make changes in the console.c file.

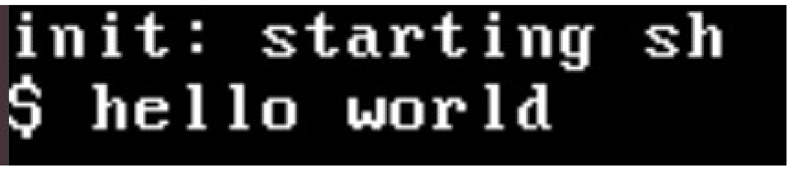
i)Moving the caret as per key pressed

For this we have to make a change in the consoleintr function in console.c . The main idea is when interrupt is generated by a LEFT/RIGHT ARROW key we appropriately change the caret position which is stored in the input.e .

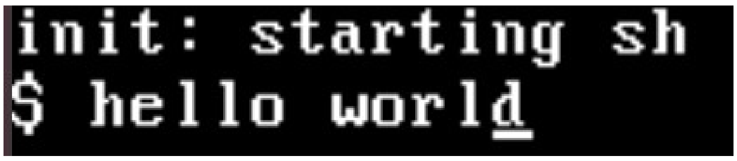


```
    case LEFTARROW:
        if (input.e != input.r) {
          input.e--;
          consputc(c);
        }
        break;
      case RIGHTARROW:
        if (input.e < input.rightmost) {
          consputc(input.buf[input.e % INPUT_BUF]);
          input.e++;
        }
        else if (input.e == input.rightmost){
          consputc(' ');
          consputc(LEFTARROW);
        }
        break;
    case UPARROW:
```

OUTPUT :
left right



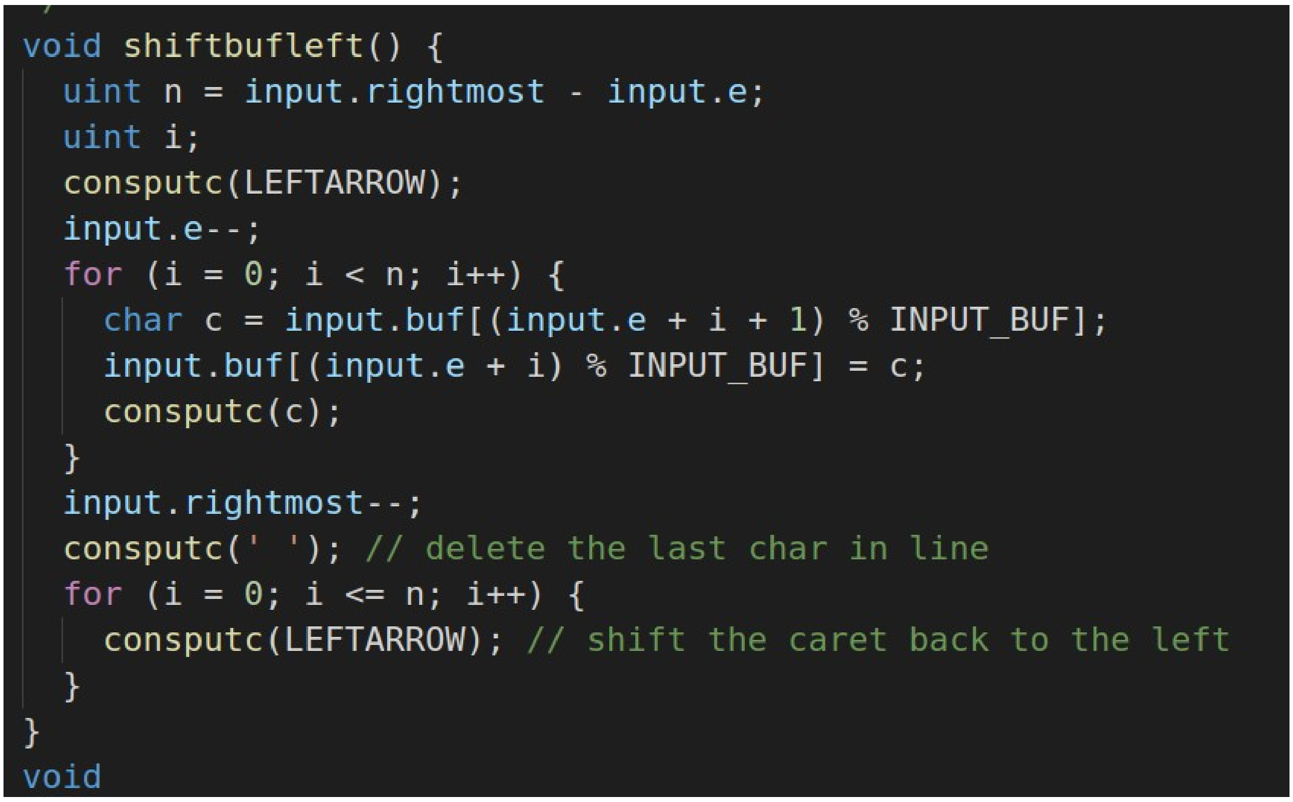
```
init: starting sh
$ hello world
```



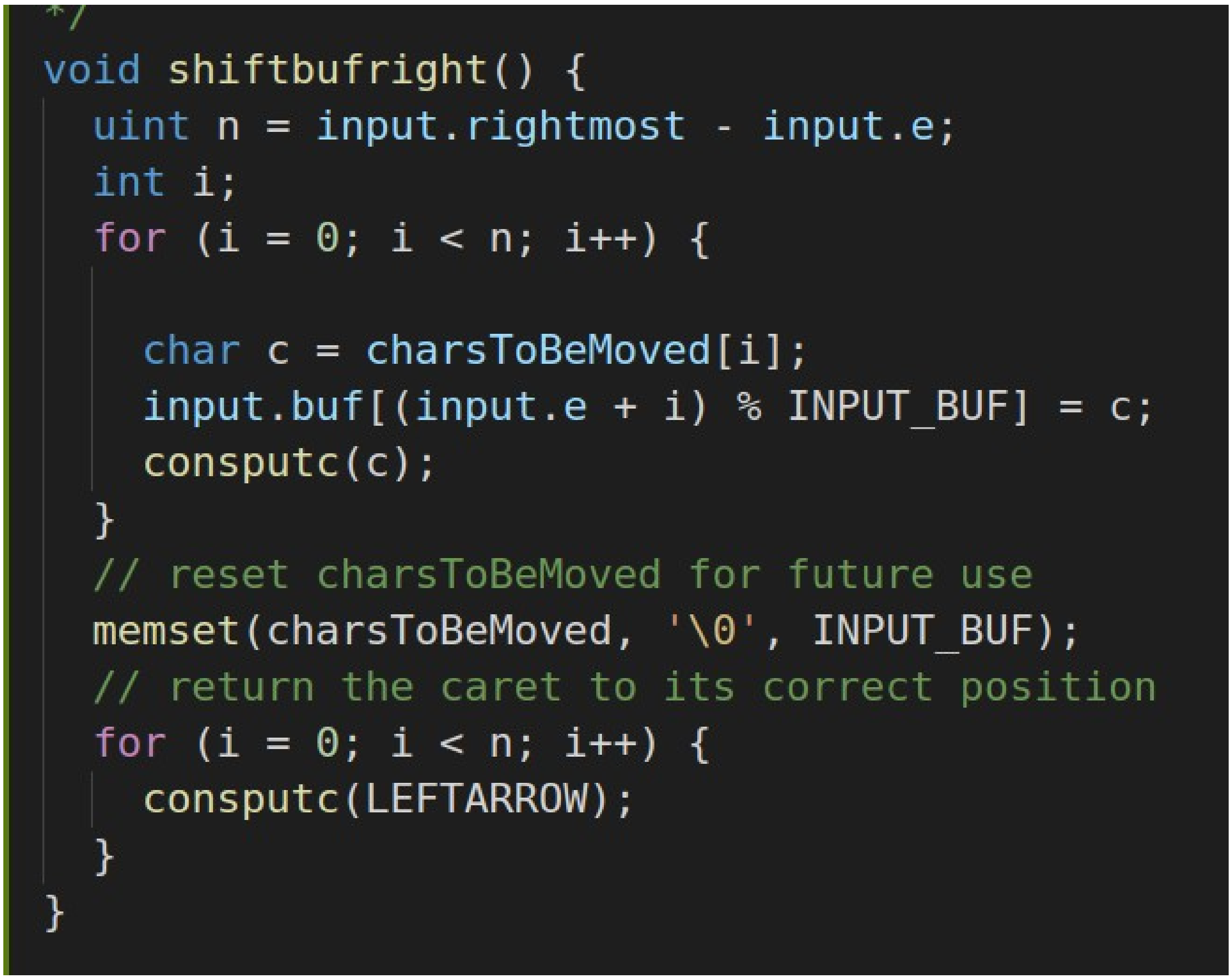
```
init: starting sh
$ hello world
```

ii) Editing of commands in the console

For this we implemented two function one for addition of charecter(shiftbufright) and one for deletion of charecter (shifbufleft) . The main idea is when we want to add/remove the element at caret then we shift the remaining of the command appropriately.



```
void shiftbufleft() {
  uint n = input.rightmost - input.e;
  uint i;
  consputc(LEFTARROW);
  input.e--;
  for (i = 0; i < n; i++) {
    char c = input.buf[(input.e + i + 1) % INPUT_BUF];
    input.buf[(input.e + i) % INPUT_BUF] = c;
    consputc(c);
  }
  input.rightmost--;
  consputc(' '); // delete the last char in line
  for (i = 0; i <= n; i++) {
    consputc(LEFTARROW); // shift the caret back to the left
  }
}
void
```



```
*/
void shiftbufright() {
  uint n = input.rightmost - input.e;
  int i;
  for (i = 0; i < n; i++) {

    char c = charsToBeMoved[i];
    input.buf[(input.e + i) % INPUT_BUF] = c;
    consputc(c);
  }
  // reset charsToBeMoved for future use
  memset(charsToBeMoved, '\0', INPUT_BUF);
  // return the caret to its correct position
  for (i = 0; i < n; i++) {
    consputc(LEFTARROW);
  }
}
```

iii)Entering to next line irrespective of the location of the caret

For this, we implemented an if statement for checking new line character ‘\n’. So when the ‘\n’ is entered then caret goes to the next line irrespective of location.

1.2

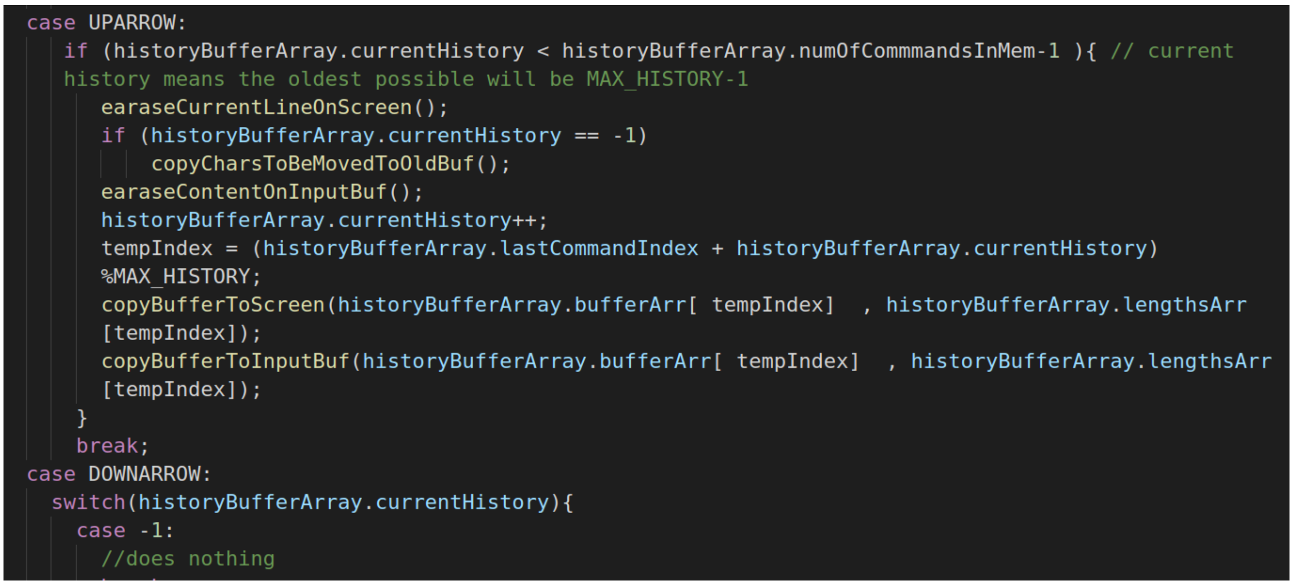
i) Implementing Shell History Ring

When we enter a command to the console the while loop in int main() of sh.c will get the command from the console and matches the input with existing commands and execute the command.

Here we have store/buffer the previsouly executed commands (max 16) and show them appropriately when UP ARROW and DOWN ARROW is pressed.

When the interrupt by the keyboard is due to UP/DOWN ARROW we will change the variable historyBuffer.placesShifted appropriately i.e, increased in case of UP ARROW and

Decreased in the other and you also see we store our previsouly executed commands in variable historyBuffer.bufferArr



```
case UPARROW:
   if (historyBufferArray.currentHistory < historyBufferArray.numOfCommmandsInMem-1 ){ // current
   history means the oldest possible will be MAX_HISTORY-1
      earaseCurrentLineOnScreen();
      if (historyBufferArray.currentHistory == -1)
          copyCharsToBeMovedToOldBuf();
      earaseContentOnInputBuf();
      historyBufferArray.currentHistory++;
      tempIndex = (historyBufferArray.lastCommandIndex + historyBufferArray.currentHistory)
      %MAX_HISTORY;
      copyBufferToScreen(historyBufferArray.bufferArr[ tempIndex]  , historyBufferArray.lengthsArr
      [tempIndex]);
      copyBufferToInputBuf(historyBufferArray.bufferArr[ tempIndex]  , historyBufferArray.lengthsArr
      [tempIndex]);
    }
    break;
case DOWNARROW:
  switch(historyBufferArray.currentHistory){
    case -1:
      //does nothing
```

ii)Adding history System call :

AS we seen in the last assignment we have to modify these files for adding a system call.

syscall.c

syscall.h- Defined the SYS_history

Sysproc.c- function SYS_history defined in sysproc.c

defs.h- declared the function “int history(char*,int);” this is called when console call the history system call.

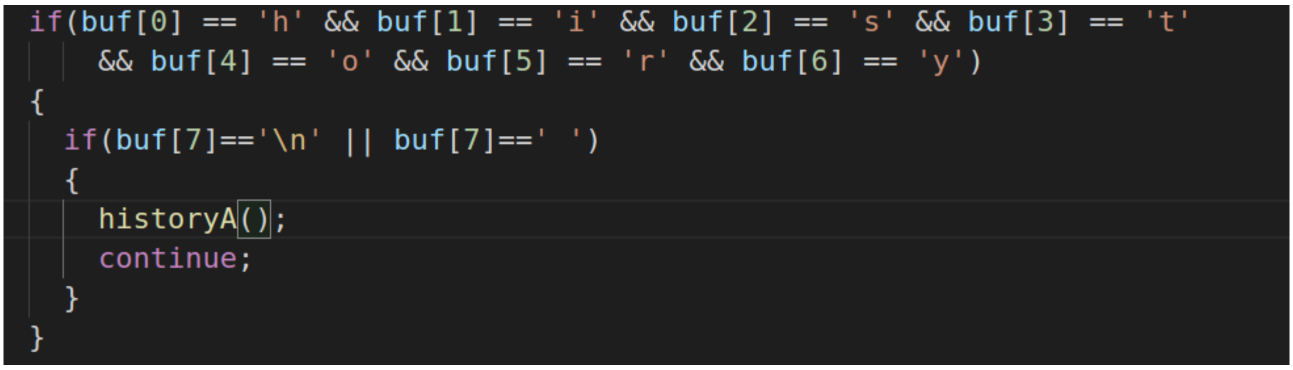
Sh.c-add history command to run history function.

Console.c- The function “int history(char*, int)” is implemented in console.c

Usys.s-Created interface “SYSCALL(history)” for a user program to call system call

iii)Adding history command :

changes are made to the sh.c file . When history command is entered showhistory function will execute.



```
if(buf[0] == 'h' && buf[1] == 'i' && buf[2] == 's' && buf[3] == 't'
    && buf[4] == 'o' && buf[5] == 'r' && buf[6] == 'y')
{
  if(buf[7]=='\n' || buf[7]==' ')
  {
    historyA();
    continue;
  }
}
```