Technical University of Republic of Moldova
Chair of Computer Science

REPORT

on OS class

Topic: Creating a command prompt that can read and parse commands

Done by: Dvorac Alexei
st. gr. FAF-101

Verified by: Lisnic Andrei

Chişinău – 2012

Topic: *Creating a command prompt that can read and parse commands*

Goals:

make the output to screen easy:

- implement putc(print a character at the current cursor position and increase the position of the cursor)

-implement puts (print a caracter at the current cursor position, and advance the position of the cursor with the length of the string)

- implement clrscr (clear the screen)

- create a command prompt which can read and parse commands, and will respond to all commands with "you typed: <command>"

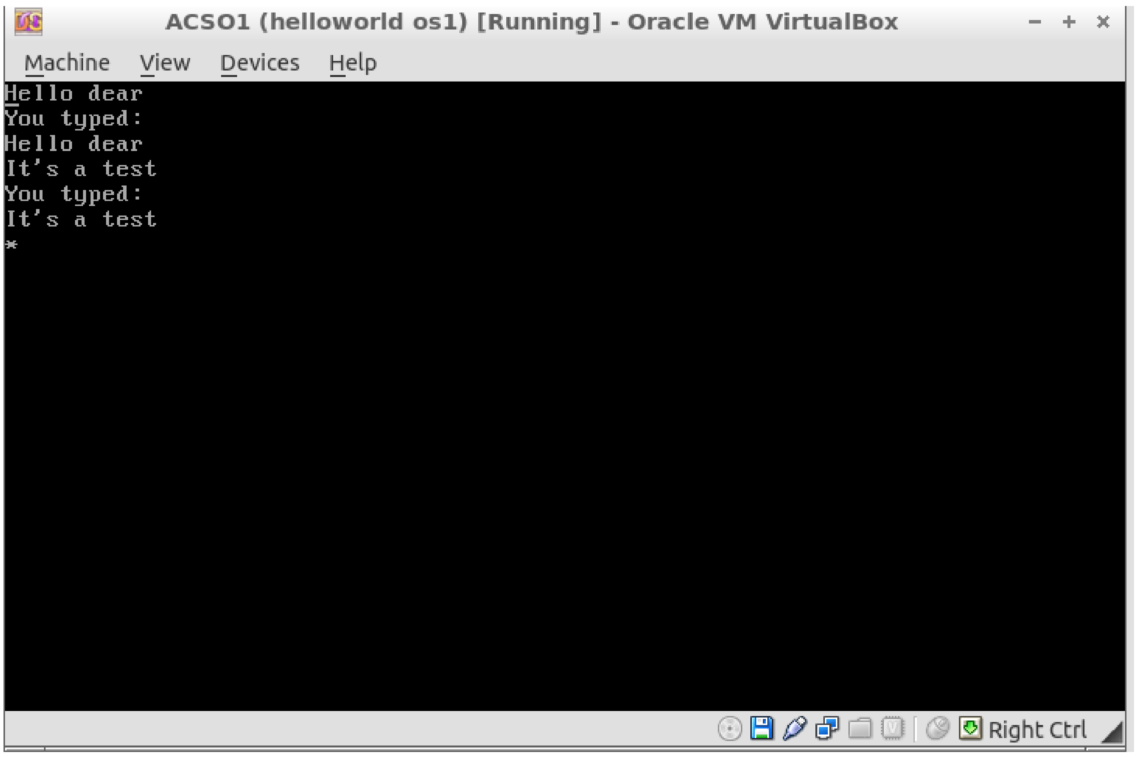
- implement gets (get a string from keyboard)

- when a command is typed, the output should be written below the command and the prompt should be moved below the output.

Workflow:

For implementing all the functions I used the getchar() function that was available. Gets, represented a sequence of getchar's and saving the data to a string. To know if the user finished the input of text, before adding the char to the string, it was compared to '\n' (Enter pressed). For putc(), was used the magic vidmem starting address. Puts – a bunch of putc() stitched together. The only difference was, that puts, as it is in C, passes on to the next line, to implement this I have just considered the number of characters in a row and added this number of spaces.

Clrscr(), simply returns to the first element of vidmem, and then fills the entire screen with spaces.


ACSO1 (helloworld os1) [Running] - Oracle VM VirtualBox
Machine View Devices Help
Right Ctrl

```
Hello dear
You typed:
Hello dear
It's a test
You typed:
It's a test
*
```

*The code is included within APPENDIX A

Conclusion: Now, our OS has some functionality, kind of a text editor. This assignment made me look for different ways to overcome the absence of various C functions/libraries.

APPENDIX A

```
#include "lib/video.c"
#include "lib/io.c"
#include "lib/kbd.c"

int i;
char* vidmem = (char *) 0xb8000;

//Here we declare putc funct, so that the puts could use it
void putc(char oneChar);

//Prints a string.Based on putc function.
void puts(char *string){
  int j;

  j=0;
  while (string[j]){
    putc(string[j]);
    j++;
  };
  NewLine();
  return;
};

//Pass on to a new line
void NewLine(){
 while(i%160!=0 || i==0){
  vidmem[i]='a';
  i++;
  vidmem[i]=0x00;
  i++;
 }
};

//print a char.
void putc(char oneChar){
  if (!i){
    i=0;
  };
  if(oneChar=='\177'){
    i-=2;
    vidmem[i]='a';
    i++;
    vidmem[i]=0x00;
    i--;
  }

  else{
    vidmem[i]=oneChar;
    i++;
    vidmem[i]=0x7;
    i++;
  }
};

void gets(char *myString){
  //char myString[256];
  char oneChar;
  int k;
  k=0;

  oneChar=getchar();
  while(oneChar!='\n'){
    putc(oneChar);
    myString[k]=oneChar;
    k++;
    oneChar=getchar();
  }
  NewLine();
};
```

```
/*****************************************/
void clrscr(){
 // char* vidmem = (char *) 0xb8000;
  i=0;

  while(i<4000){
    vidmem[i]='a';
    i++;
    vidmem[i]=0x00;
    i++;
  }
  i=0;
};
/**************************************/

void commandPrompt(myString){
    gets(myString);
    if (myString=='clrscr'){
      clrscr();
    }
    else{
      puts("You typed: ");
      puts(myString);
    }
};

int main( void )
{

  char someNewChar;
  char myString[256];
  int commandCounter;

  commandCounter=0;

  while(1){
    if(commandCounter%5==0){
      clrscr();
    }
  commandPrompt(myString);
  commandCounter++;
}

}
```