John Brisebois
OS ReadMe

**Commands:**

| | |
|---|---|
| new <folder name> | Creates folder in current. Example: “new documents” |
| cd <folder name> | Goes to child folder. Example: “cd documents” |
| cd . | returns to parent folder |
| delete <folder name> | Deletes a child folder. Example: “delete documents” |
| dir | show all folder children |

**How to Use:**

The user can use the above commands to create and manipulate folders. There is no data that can be stored in said folders besides the name and child folders. At startup there is only a home directory, and the user must create anything else themselves.

**Lines of Code:**

There are a little over 400 lines of code in this OS, most of which are in kernel.c. Keep in mind I space my code out a fair amount, so it may be closer to 350.

**Help and References:**

- The Little Book about OS Development Erik Helin, Adam Renberg
- Create Your Own Operating System Malshani Dahanayaka
- How to Write a Simple Operating System Mike Saunders, MikeOS Developers

Mostly, I used “The Little Book about OS Development” with help from “How to Write a Simple Operating System,” which is based heavily on the former with some tips and tricks for modification. Much of the makefile, bootloader, io interface, and framebuffer interface are copied from these texts (with modifications), although they add up to a small amount of the total code. I used “How to Write a Simple Operating System” to learn some of the basics, but none of the code in this OS is pulled from it.

**Issues and Compromises:**

Unfortunately, I was unable to get my linker script running correctly while trying to set up virtual memory. Also, I failed in getting my OS to read and write to the hard disk, meaning it is reset after shutdown. Because of these two factors I was forced to use arrays to store folder data, leading to severe limitations if I wanted to expand the OS.

Bios calls didn’t seem to work either, although they worked fine for me before in assembly based kernels, so the way in which I interact with the keyboard is suboptimal and leads to lag.

I was unable to get scrolling in the command line to work, so there is currently no way to see command history as everything stays in one place on the screen.

It was extremely difficult to troubleshoot issues for two reasons. First, oftentimes the OS would simply fail to boot without any error codes, leading to a lot of trial and error without proper knowledge of what’s wrong. Second, online help is very difficult to find since everyone’s project differs greatly and only helped me a couple of times.

John Brisebois
OS ReadMe

Because the folder system was a last minute compromise to replace writing to disk or using nodes (no malloc or free without virtual memory), it can be broken quite easily by the user. The result is folders pointing to incorrect children.

**How the Kernel Works:**

The kernel runs through a while loop while reading input from the keyboard. It then segments the inputted string into a command and an object. If the command is recognized it executes it and returns to the loop.

Folders are kept in memory using two arrays: one storing their names and one storing their children and parent indexes. There can be up to 100 total folders, max 10 children per folder, and a name limit of 10 characters. This is suboptimal, but due to the issues with disk writing and virtual memory it is the best I could do. File types could be added with some modifications to the array data, but unfortunately I didn't have the time.

**Running the OS:**

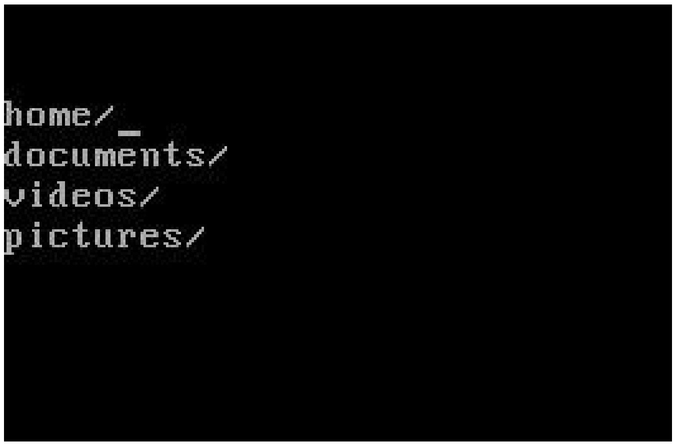
The OS can be run on linux with Bochs using the command make run. The following packages are required: build-essential, nasm, genisoimage, bochs, and bochs-sdl. Getting it to compile and run with bochs on my end required a lot of tweaks to build versions and such, but if you can't get it to run simply use Virtual Box to run os.iso.

**Functions:**

- **loader.s**: sets up the sections, constants, stack, and calls the kernel
- **io.s:**
  - **outb**: sended byte to io port
  - **inb**: receives byte from io port
- **kernel.c:**
  - **fb_move_cursor**: sends inputted position to the port that holds the location of the cursor.
  - **fb_write_cell**: send two bytes to the framebuffer array containing a char and two colors.
  - **fb_write**: use the above function to print a whole string to the screen
  - **getCode**: use inb to get an input from the keyboard
  - **getAscii:** convert a keyboard code to a char
  - **newLine:** move cursor down a row
  - **compareStr:** compare two string against each other
  - **compareNames:** compare a string against a folder name (in the form of an array index)
  - **resetStr:** set every char in a string to " "
  - **clearLine:** set every char in a row on screen to be blank
  - **clearFiles:** set all file names to be blank, and all pointer info to -1

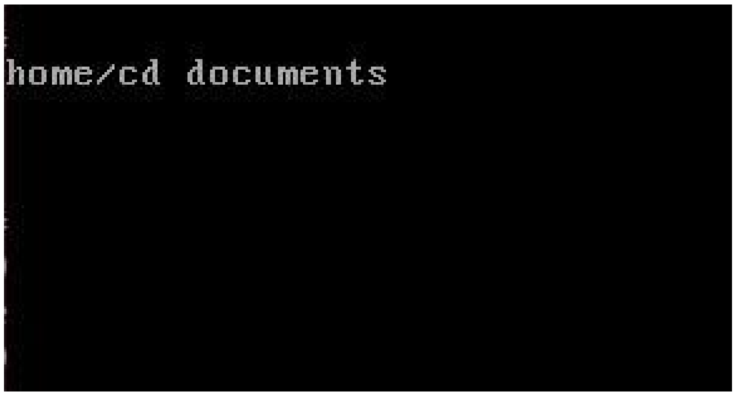
- **writeFolderName:** print a folders name to screen
- **exeDir:** executes upon the command dir.  Prints all children to screen
- **clearSpace:** clears the space on screen that dir writes to
- **writeDir:** prints current directory and all parents to screen
- **createFolder:** creates a new folder as a child of current directory
- **goToParent:** set current directory to parent.  Executes upon the command cd .
- **goToChild:** set current directory to a specified child.  Executes upon cd <name>
- **deleteIndex:** deletes the folder at a specified index
- **delete:** converts a string to an index and calls deleteIndex
- **parseCommand:** segment an input string into a command and an object
- **exe**: compare inputted command against all recognized ones, then runs the appropriate function.
- **kernel_main:** kernel entry point.  loops while polling the keyboard and modifies input array accordingly.

**Demo:**



```
home/_
documents/
videos/
pictures/
```

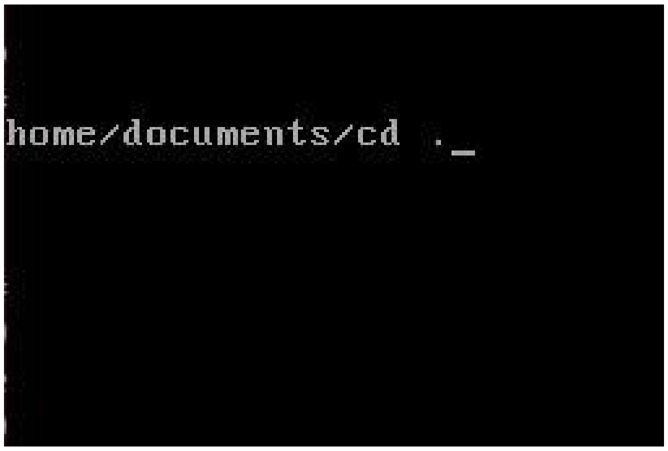
User has added folders documents, videos, and pictures, which are displayed with dir



```
home/cd documents
```

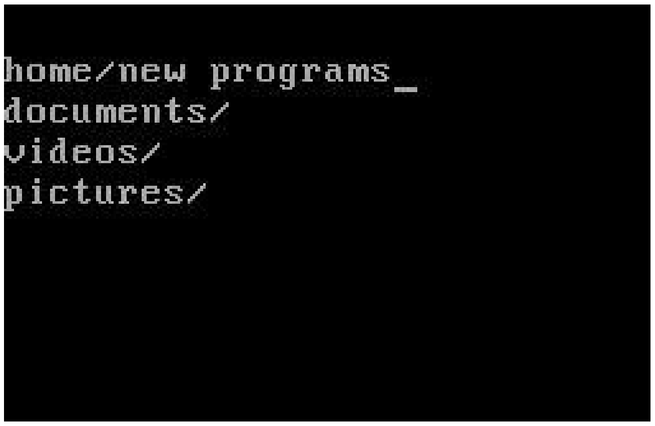
Go to documents

John Brisebois
OS ReadMe



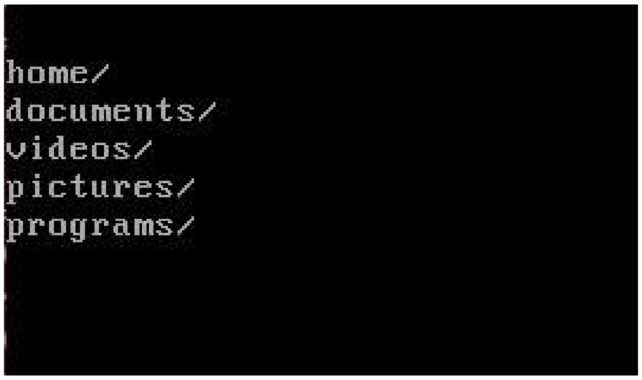
```
home/documents/cd ._
```

While in documents, return to parent



```
home/new programs_
documents/
videos/
pictures/
```

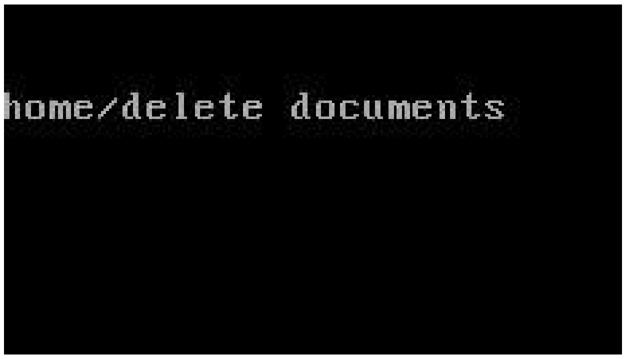
Create folder programs



```
home/
documents/
videos/
pictures/
programs/
```

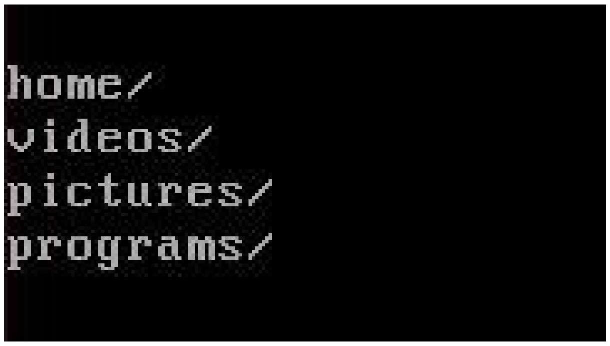
Display dir, now with programs

John Brisebois
OS ReadMe



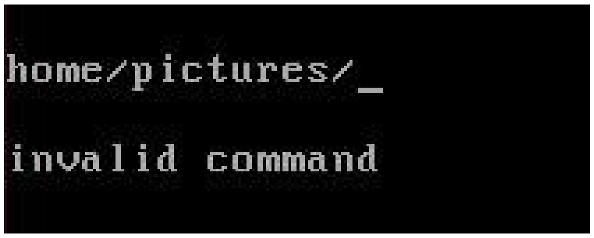
```
home/delete documents
```

Delete documents folder



```
home/
videos/
pictures/
programs/
```

Documents is gone



```
home/pictures/_

invalid command
```

Invalid command notice