11/2/2017

**Submitted To**

**Md. Saeed Siddik**

**Lecturer, IIT, University of Dhaka**

**Submitted by**

**Team Odin**

**BSSE 0807 Mahir Mahbub**

**BSSE 0815 Shuvradeb Saha**

**BSSE 0822 S.M. Khayrul Islam**

**BSSE 0830 Obaidur Rahman**

**BSSE 0839 Sefat-E-Mahadi**

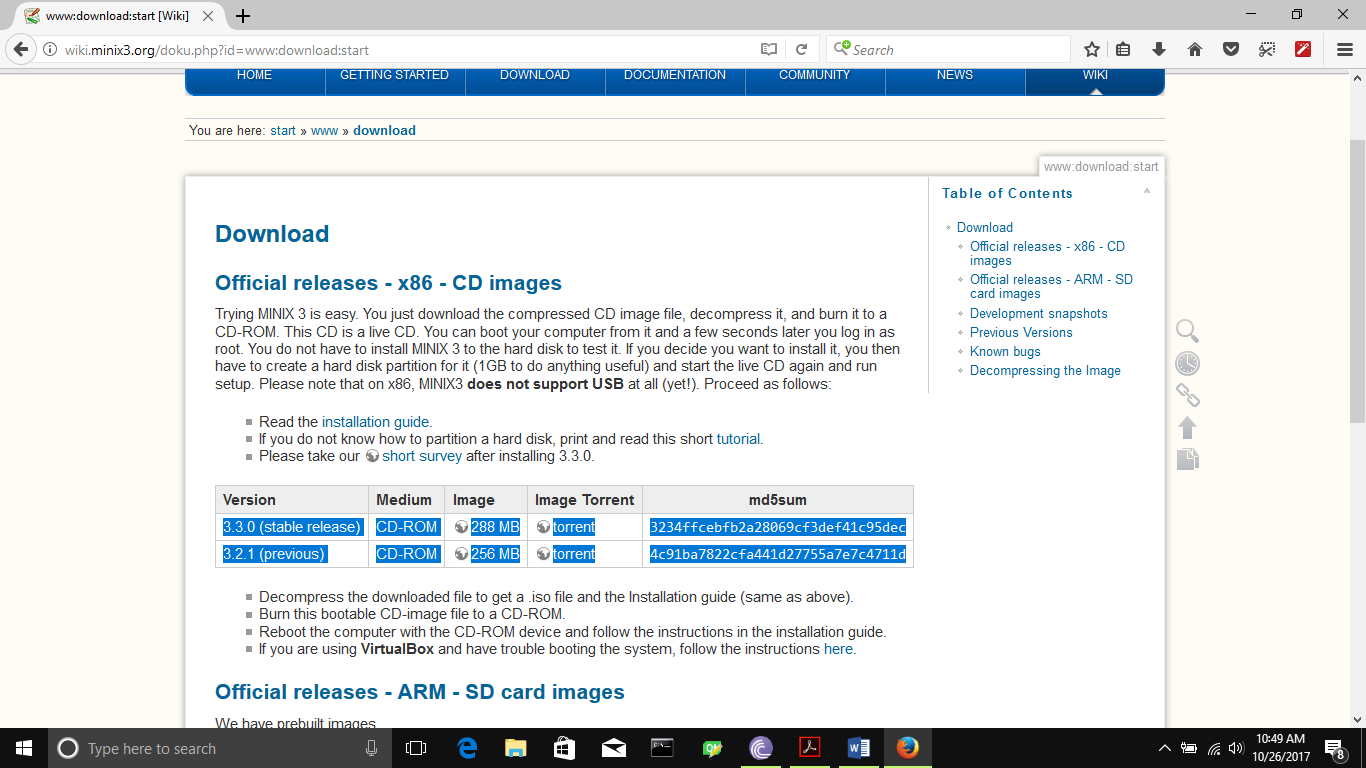
**CSE 401- Operating System & System Programming**

**OS Lab Project Minix**

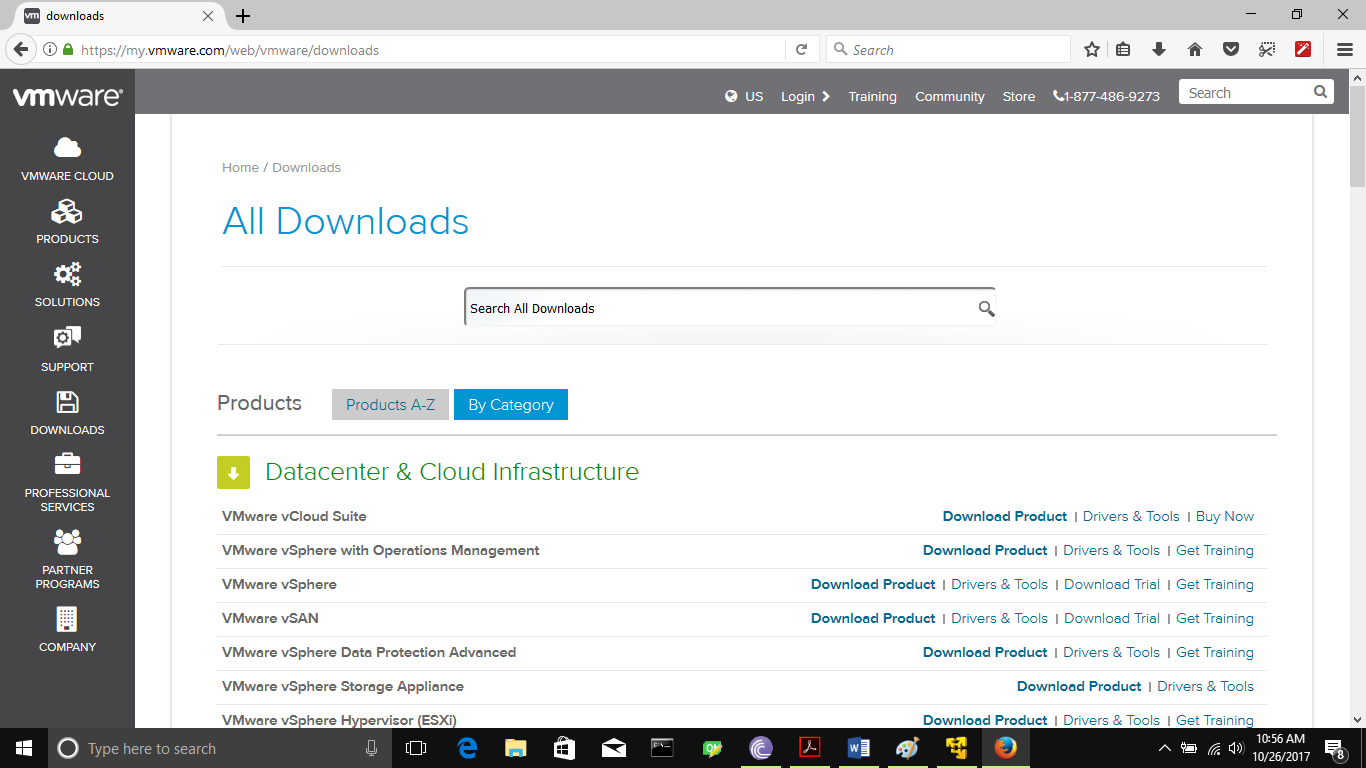
**Question 1:**

**Requirement 1: MINIX** (from "mini-Unix") is a Unix-like computer operating system based on a microkernel architecture. MINIX 3 was publicly announced on 24 October 2005 by Andrew Tanenbaum during his keynote speech on top of the ACM Symposium Operating Systems Principles conference. Although it still serves as an example for the new edition of Tanenbaum and Woodhull's textbook, it is comprehensively redesigned to be "usable as a serious system on resource-limited and embedded computers and for applications requiring high reliability.

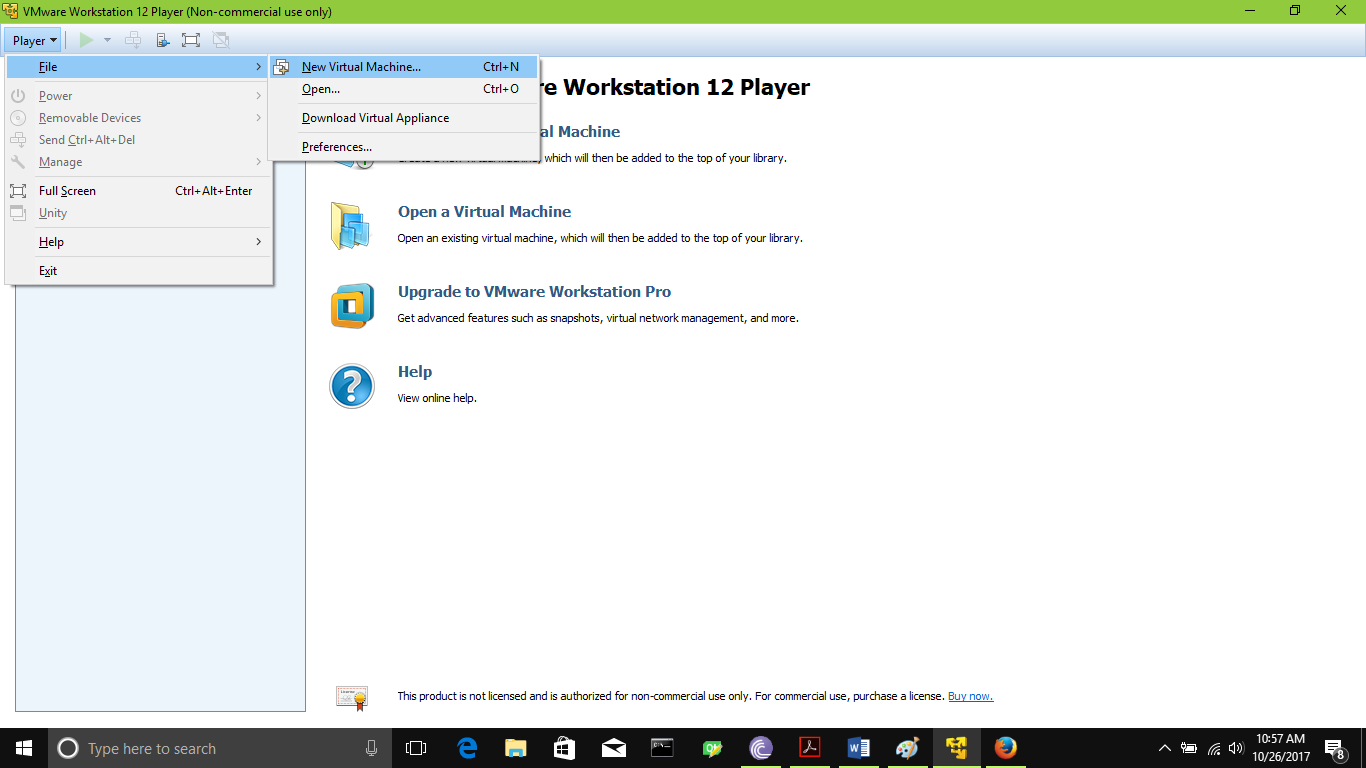
Here we explain the installation process of Minix 3 in VMware. First visit the <http://www.minix3.org>to download the .iso file.



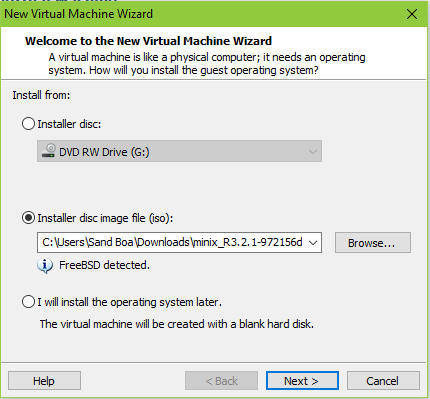
Then download VMware from its official website. <http://www.my.vmware.com>

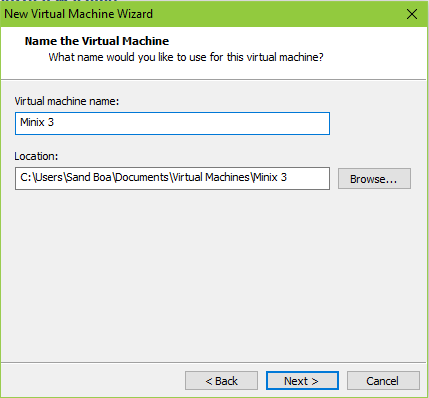


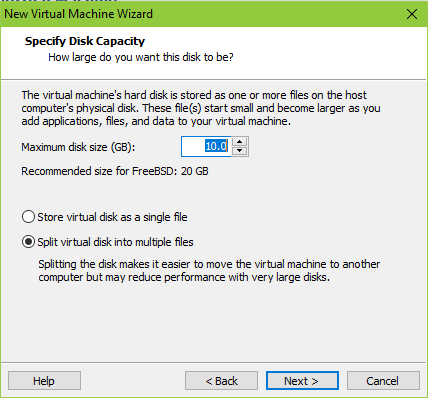
Then install Vmware and open it.Select Player->New Virtual Machine

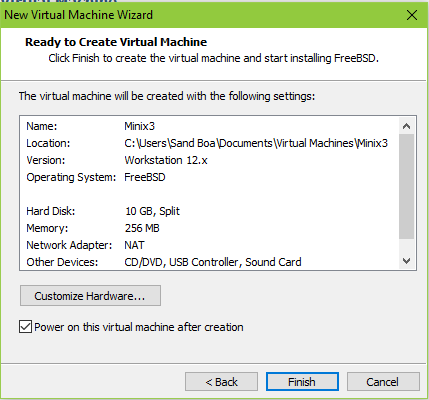


Then follow the instruction in Pictures below.

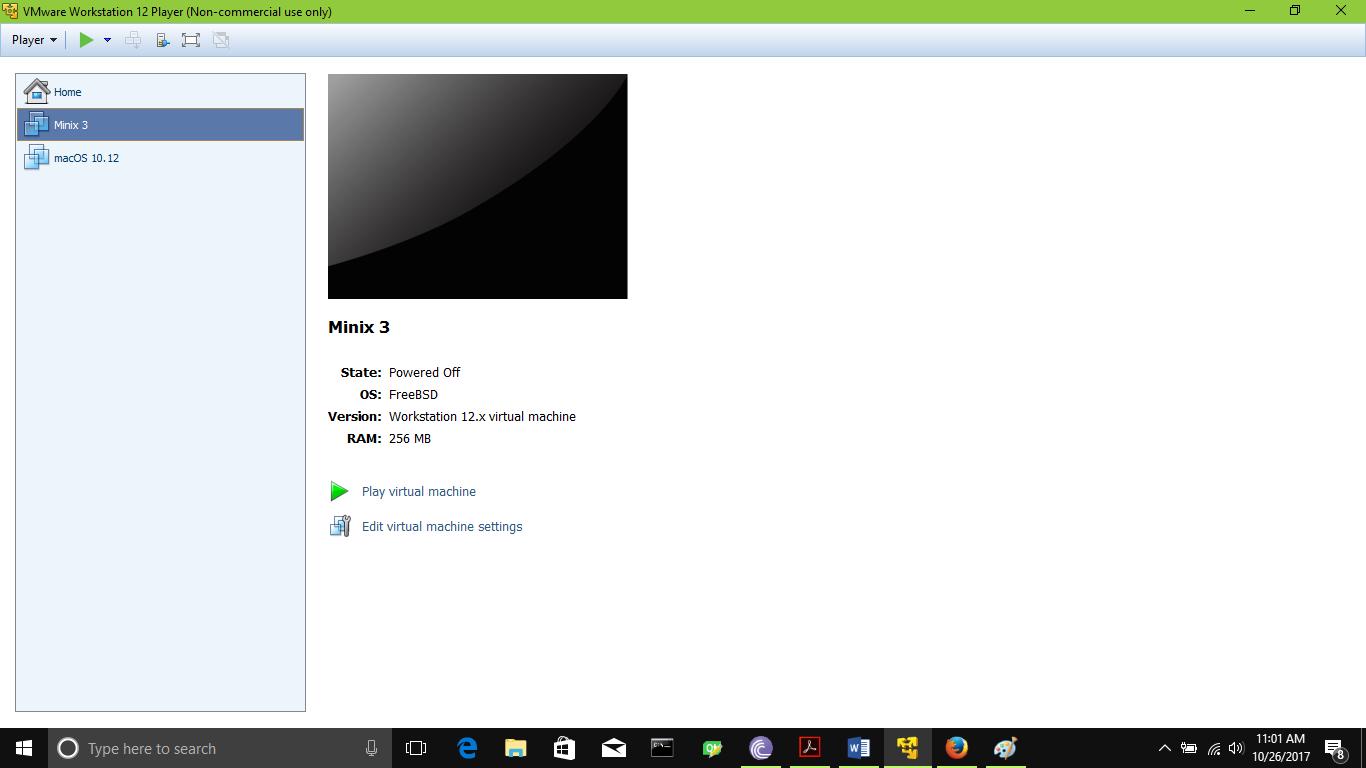




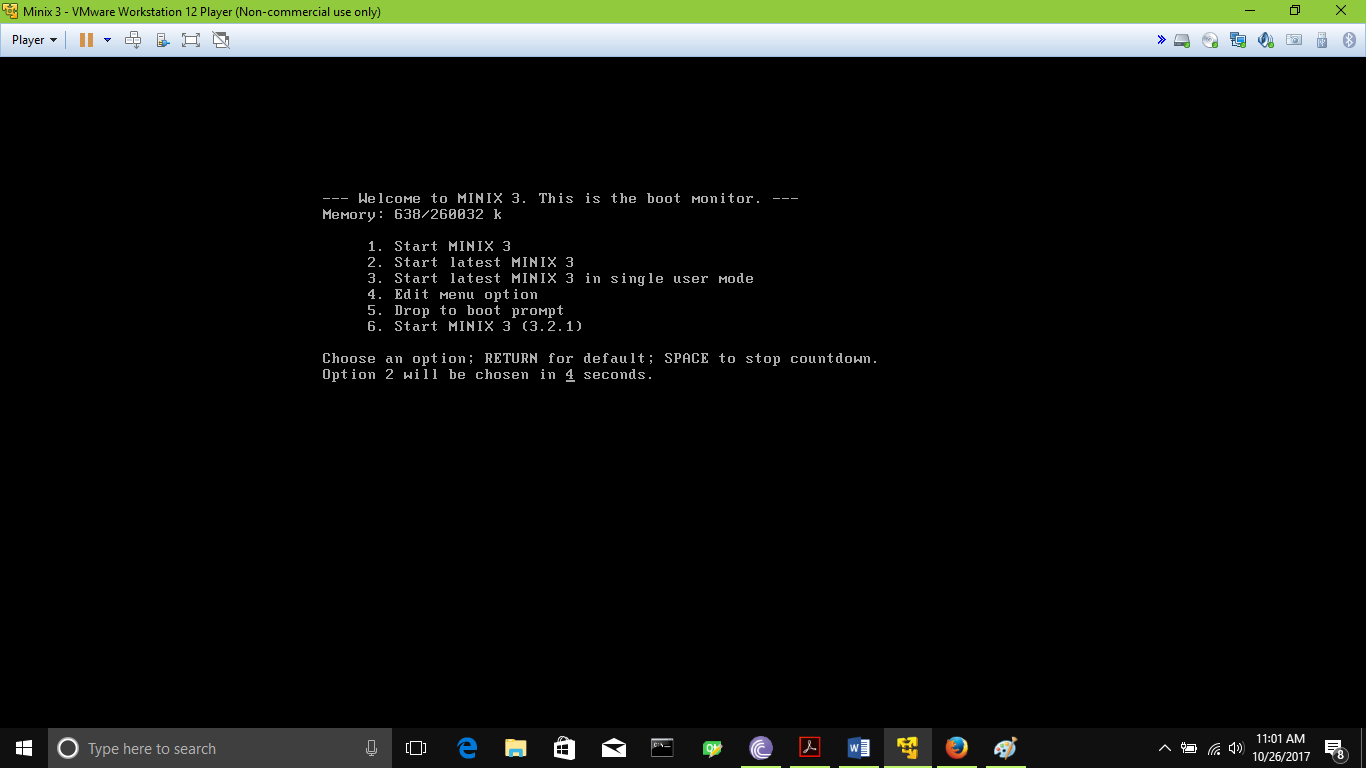




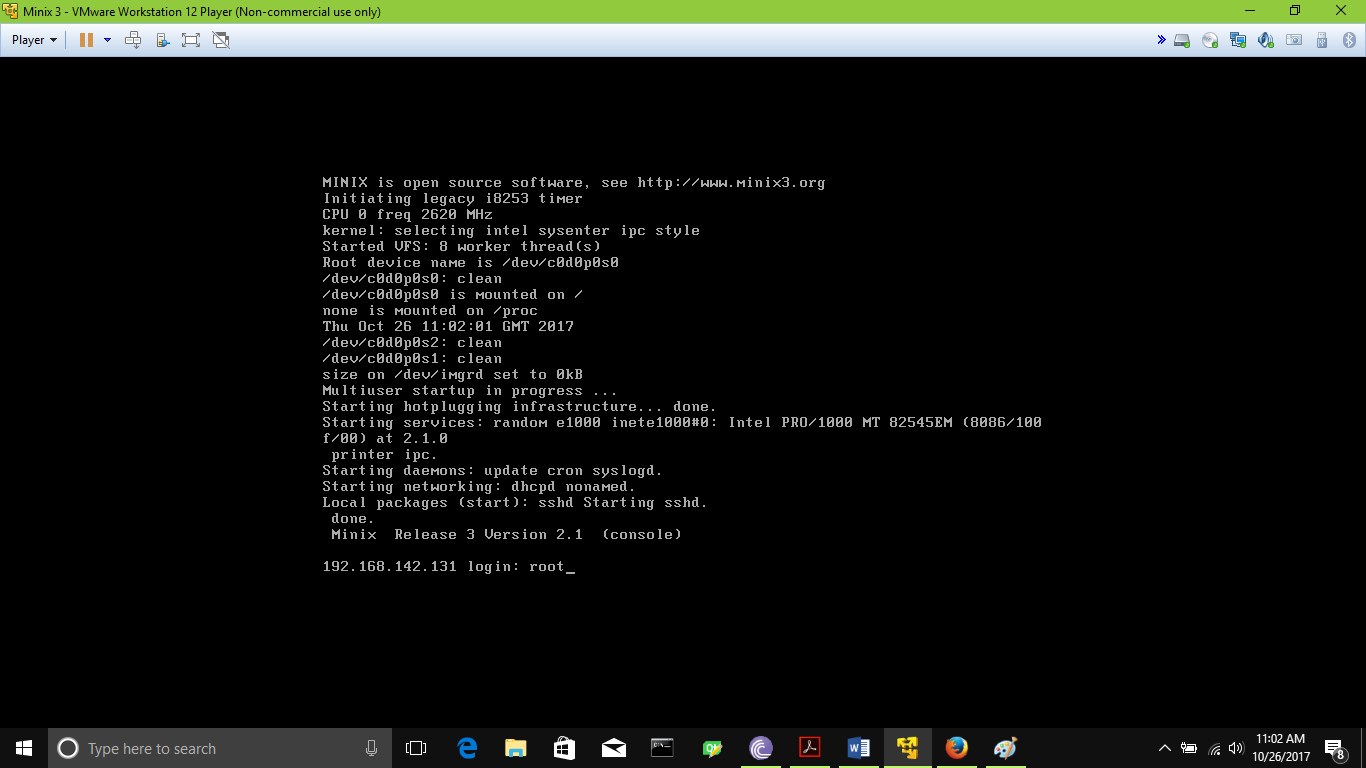
When VMware image creation is complete, Open the Minix in VMware.



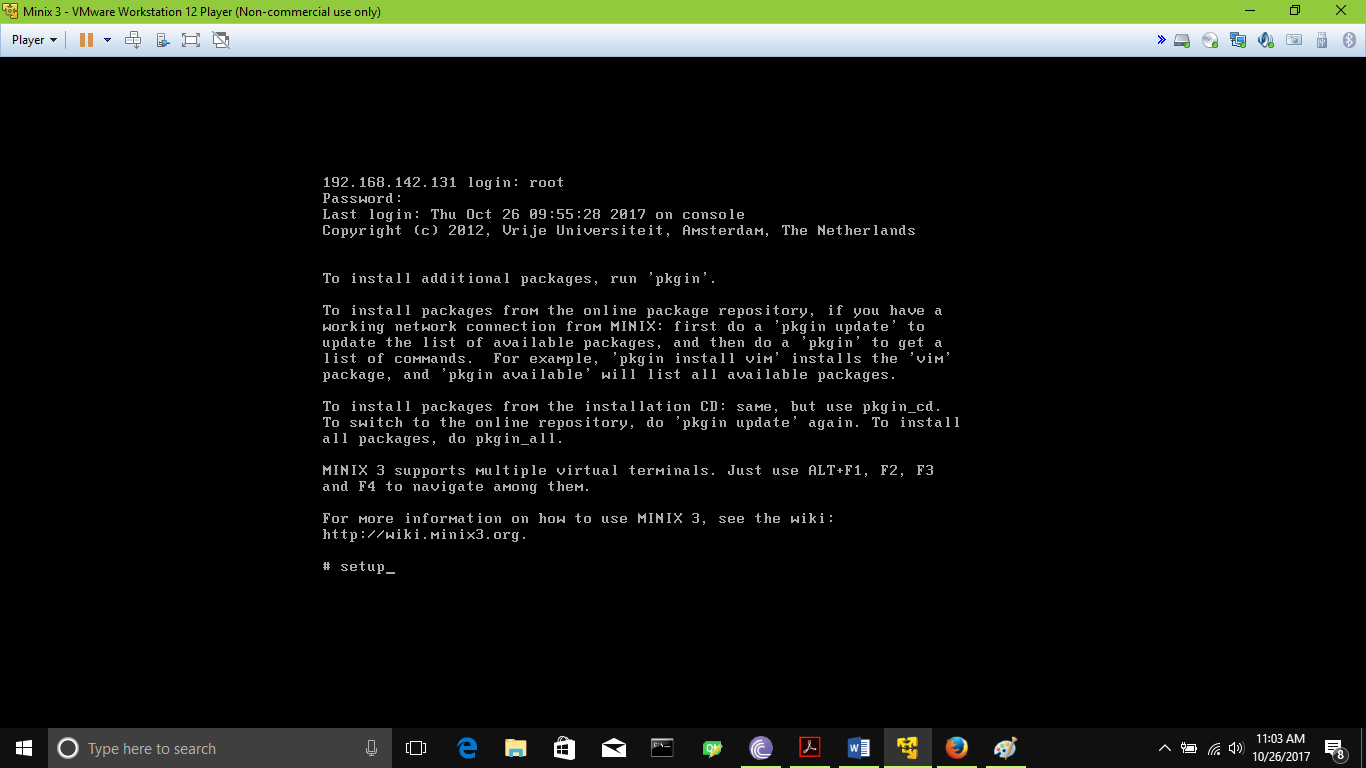
Select the option to open the Minix. Basically select 1.

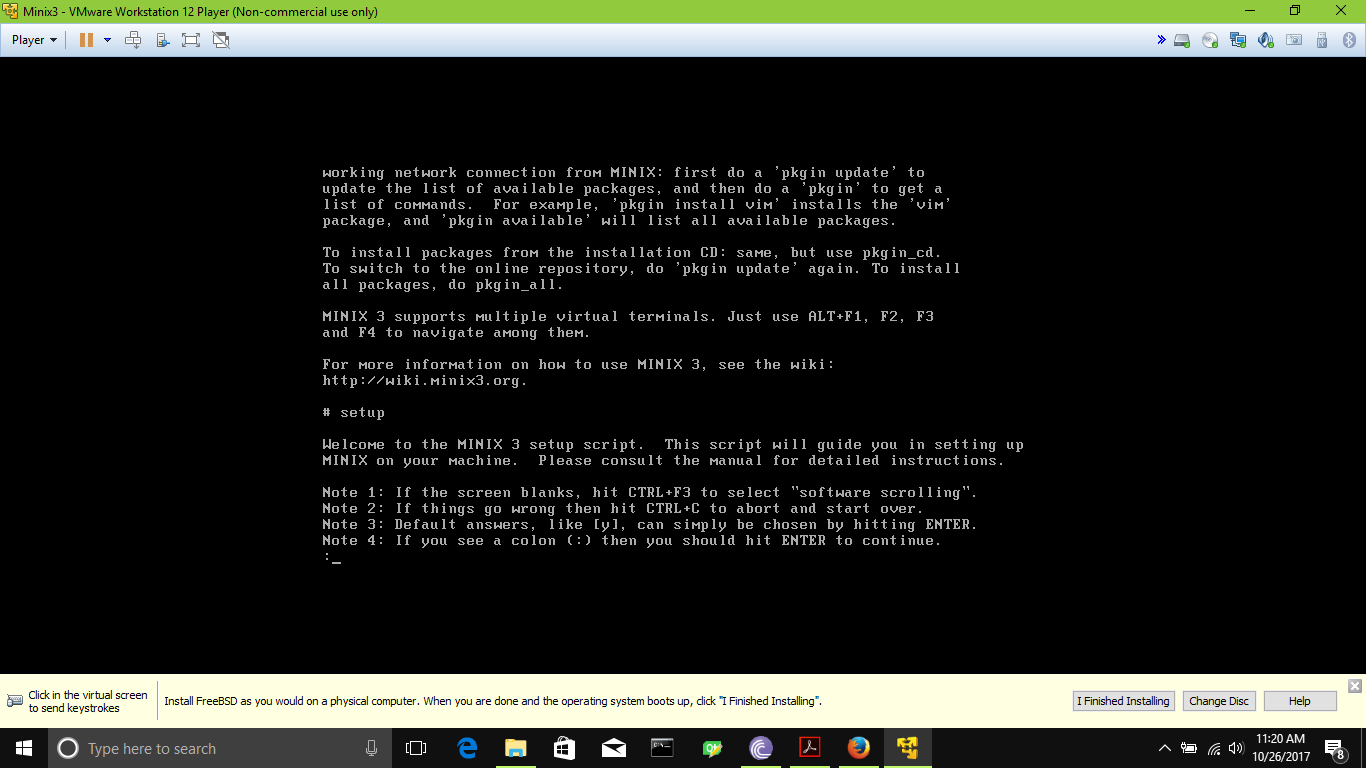


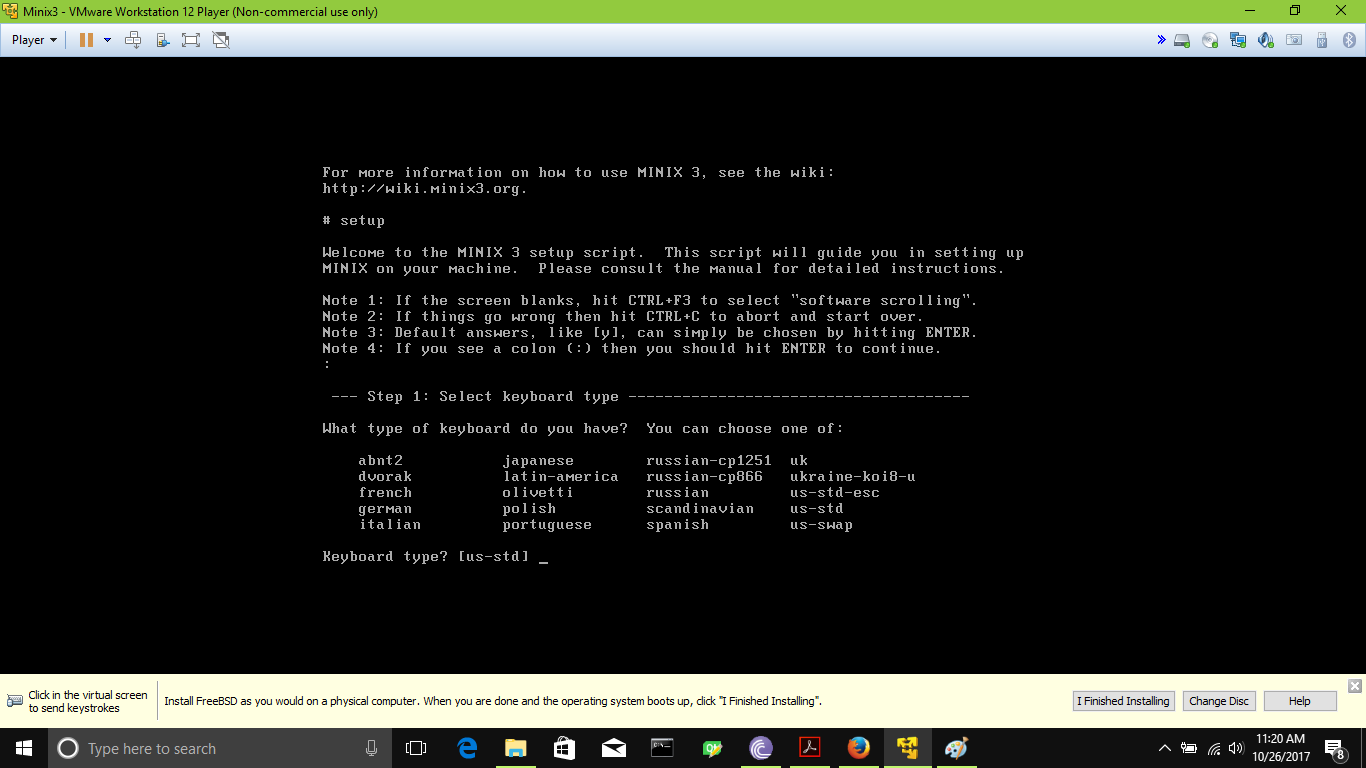
Log in with root .

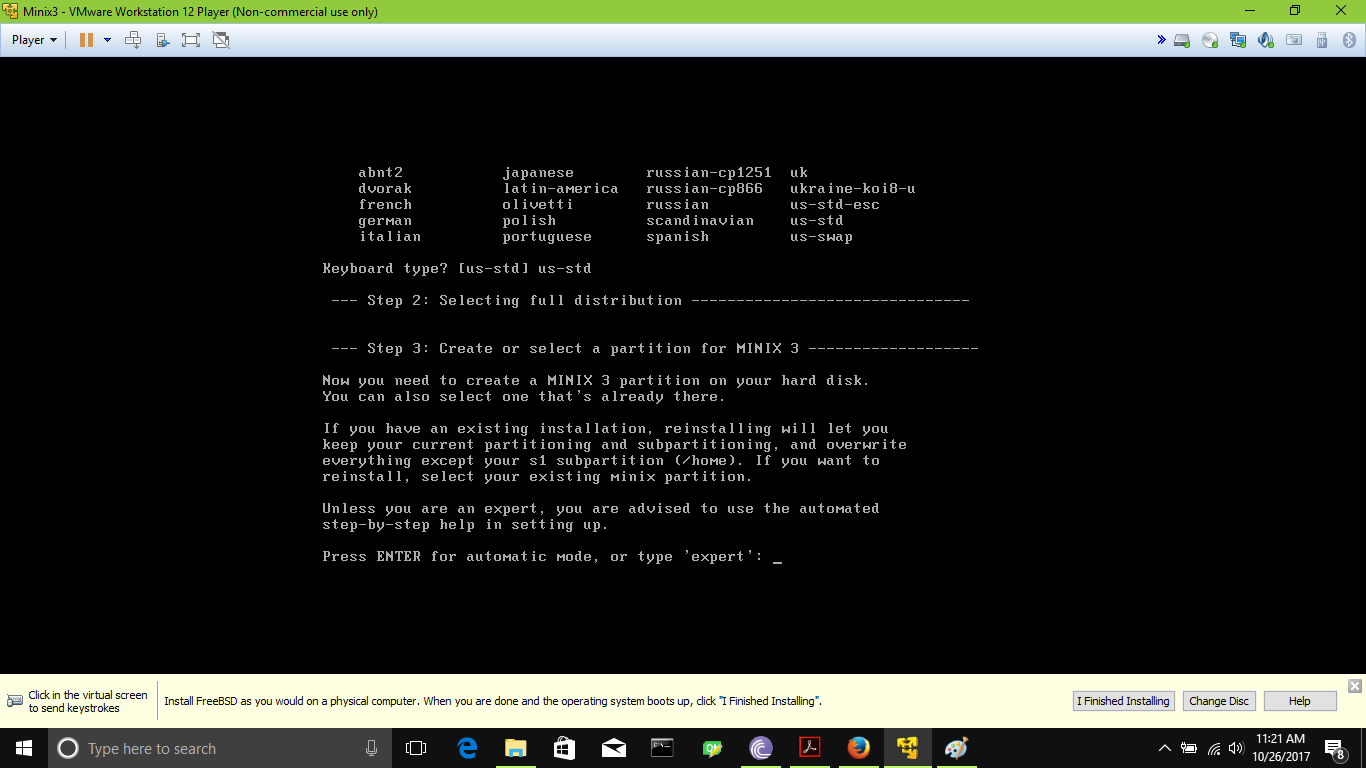


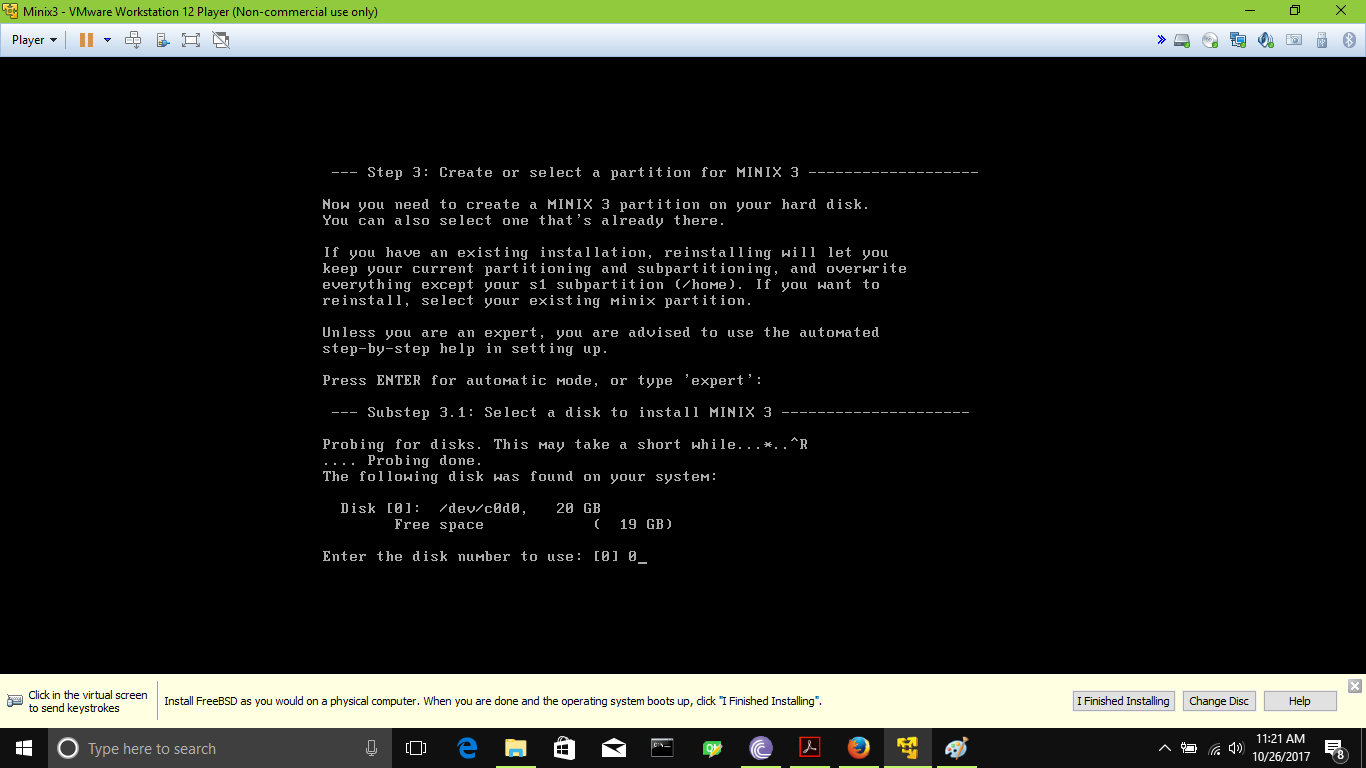
Enter command “Setup” and select keyboard and DHCP if available.Follow the pictures.

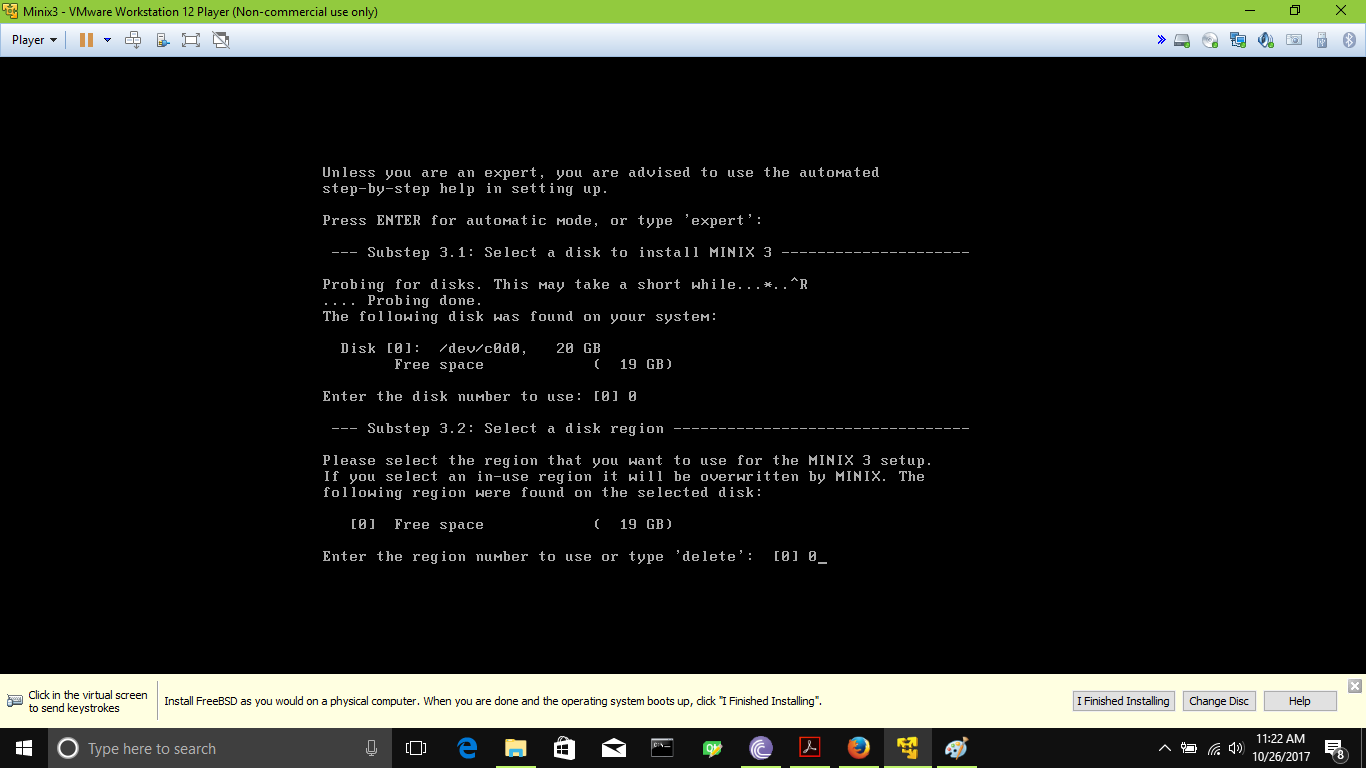


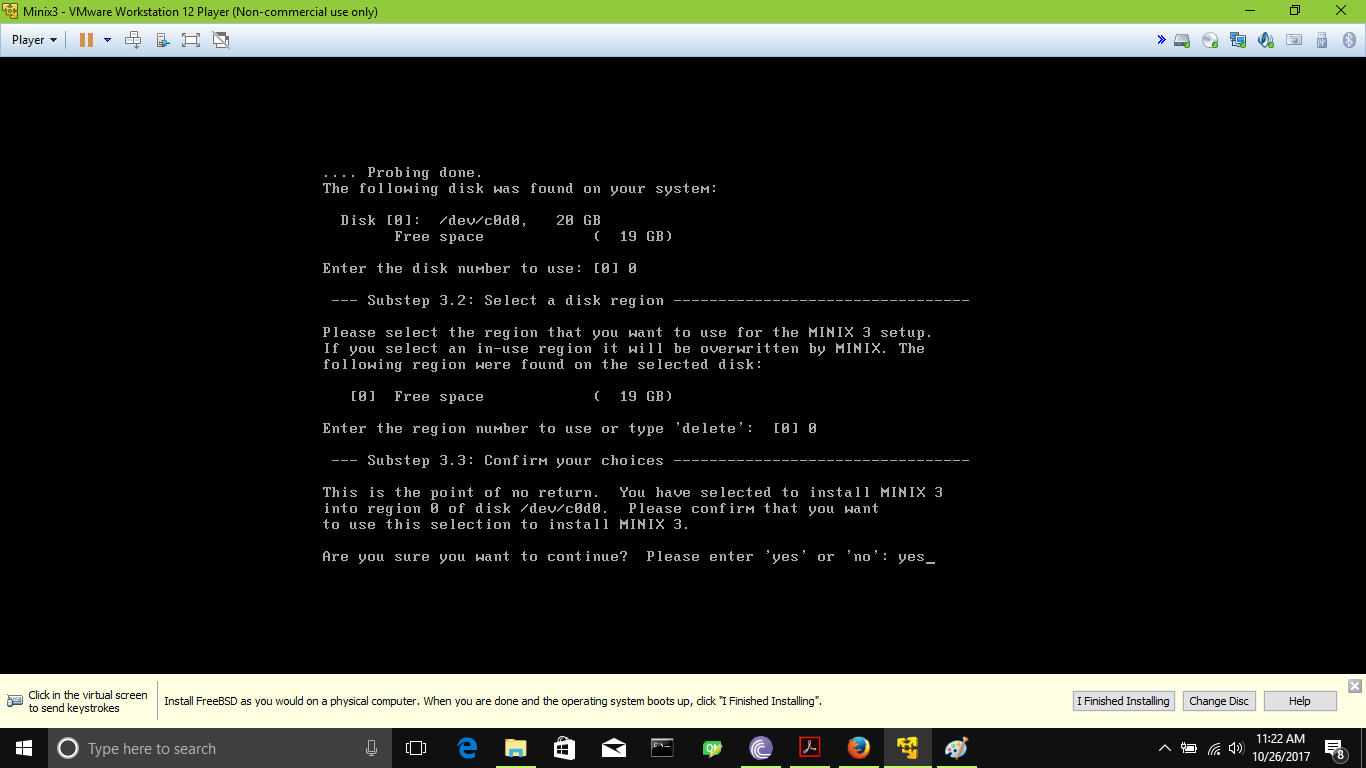


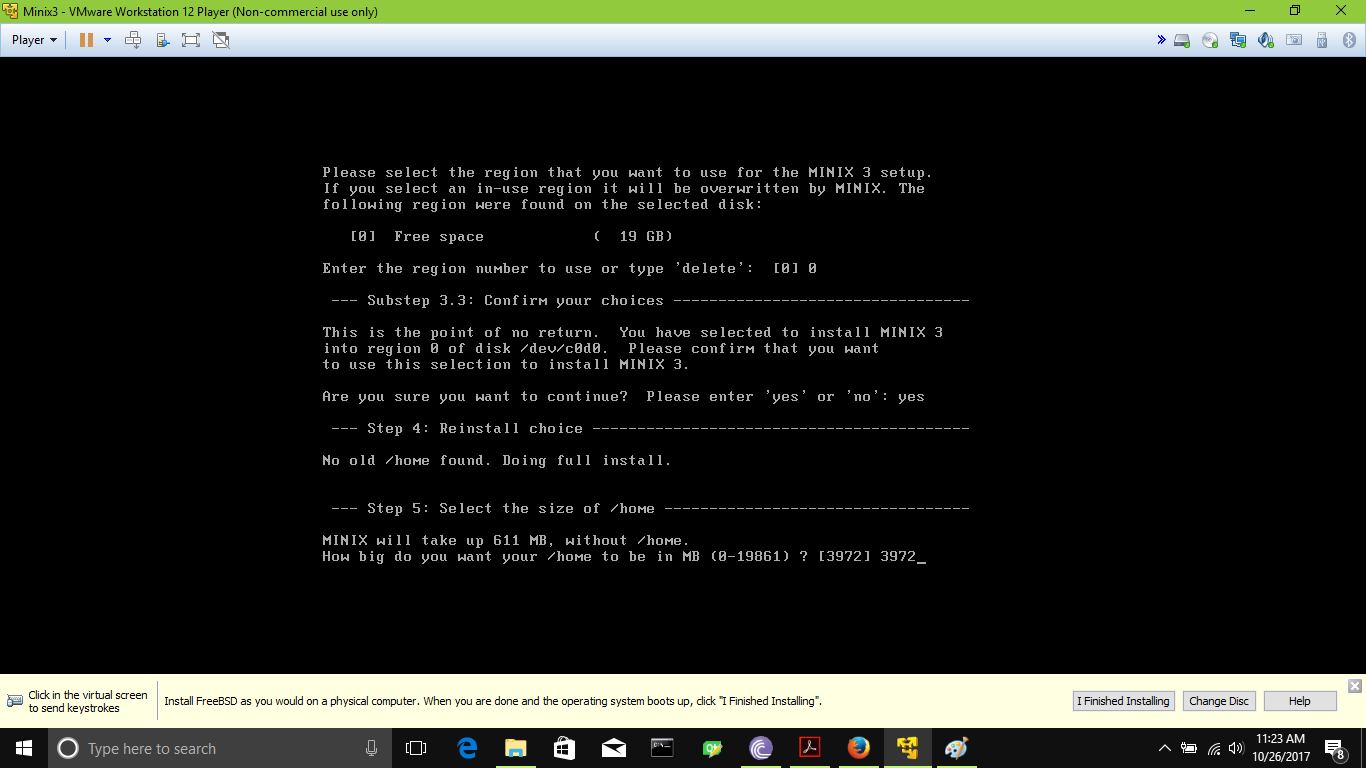


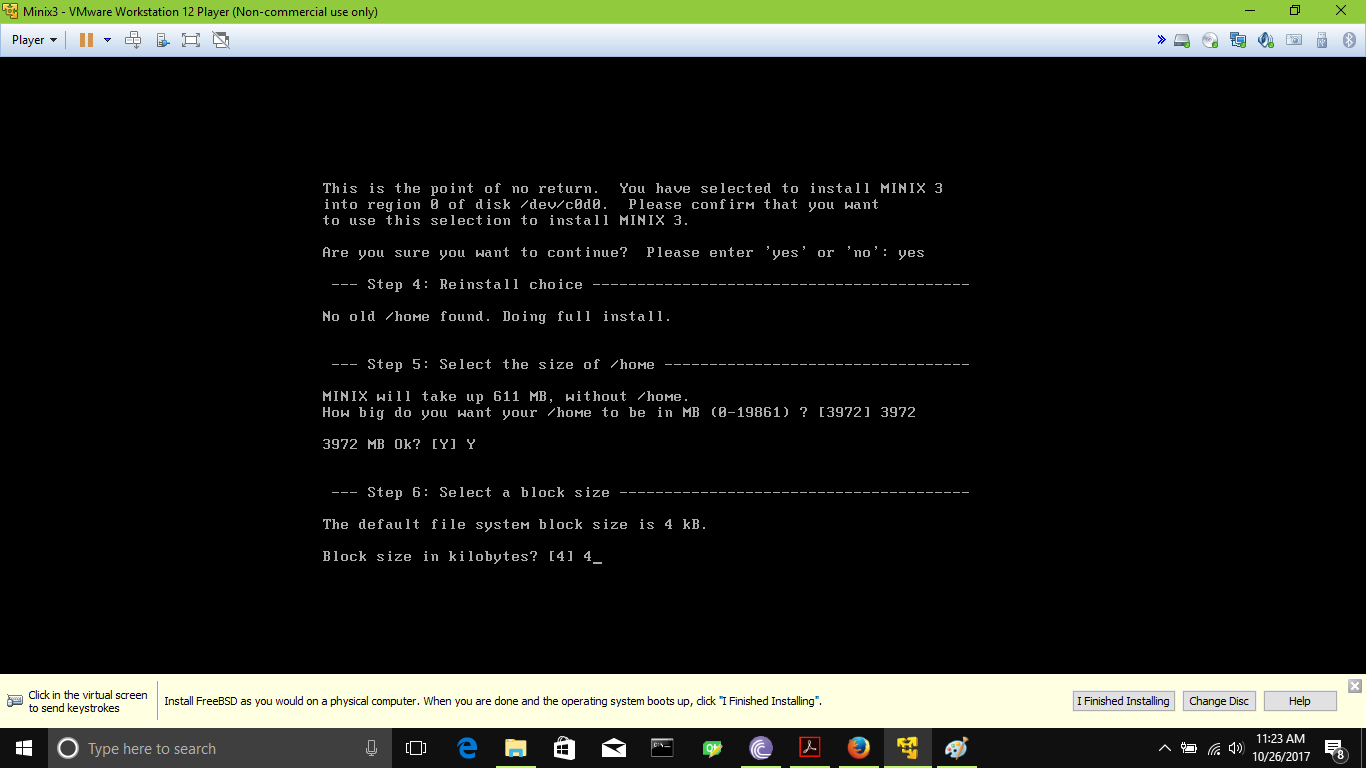


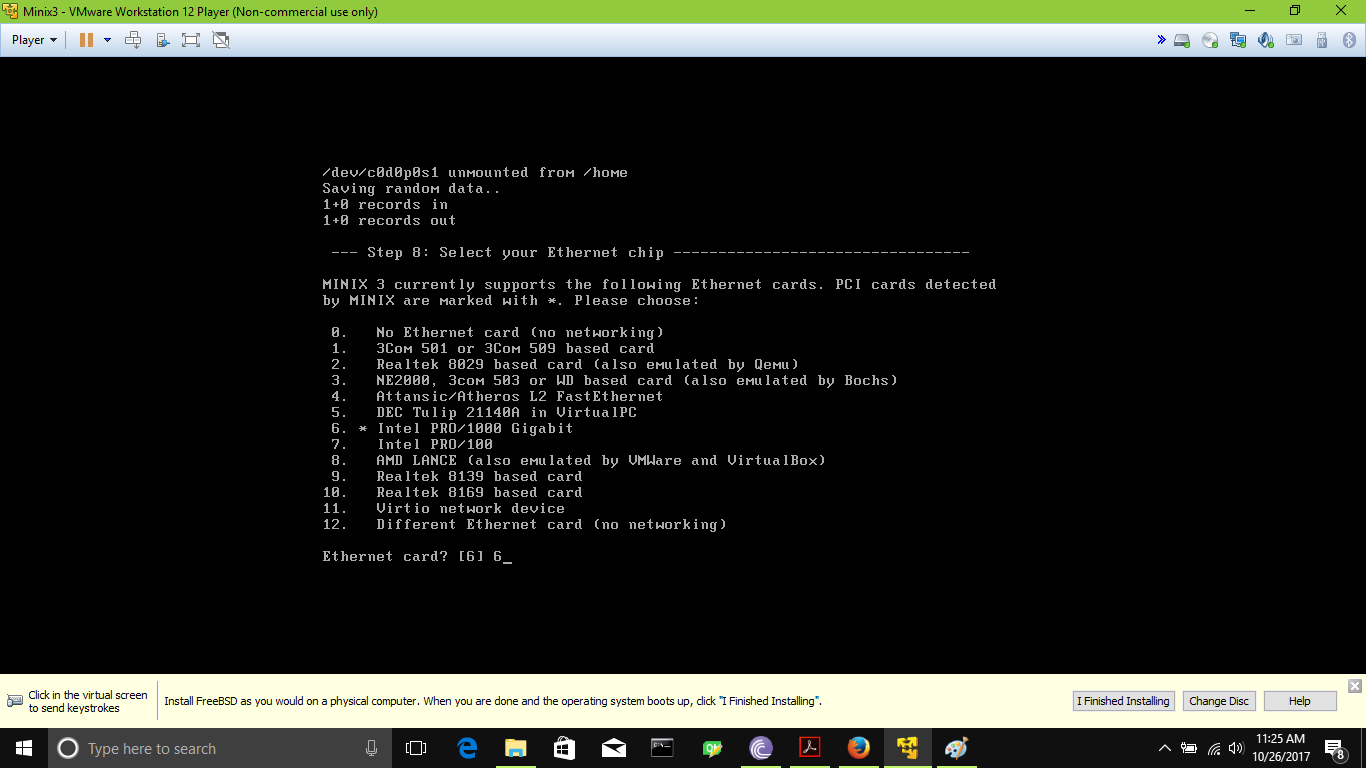


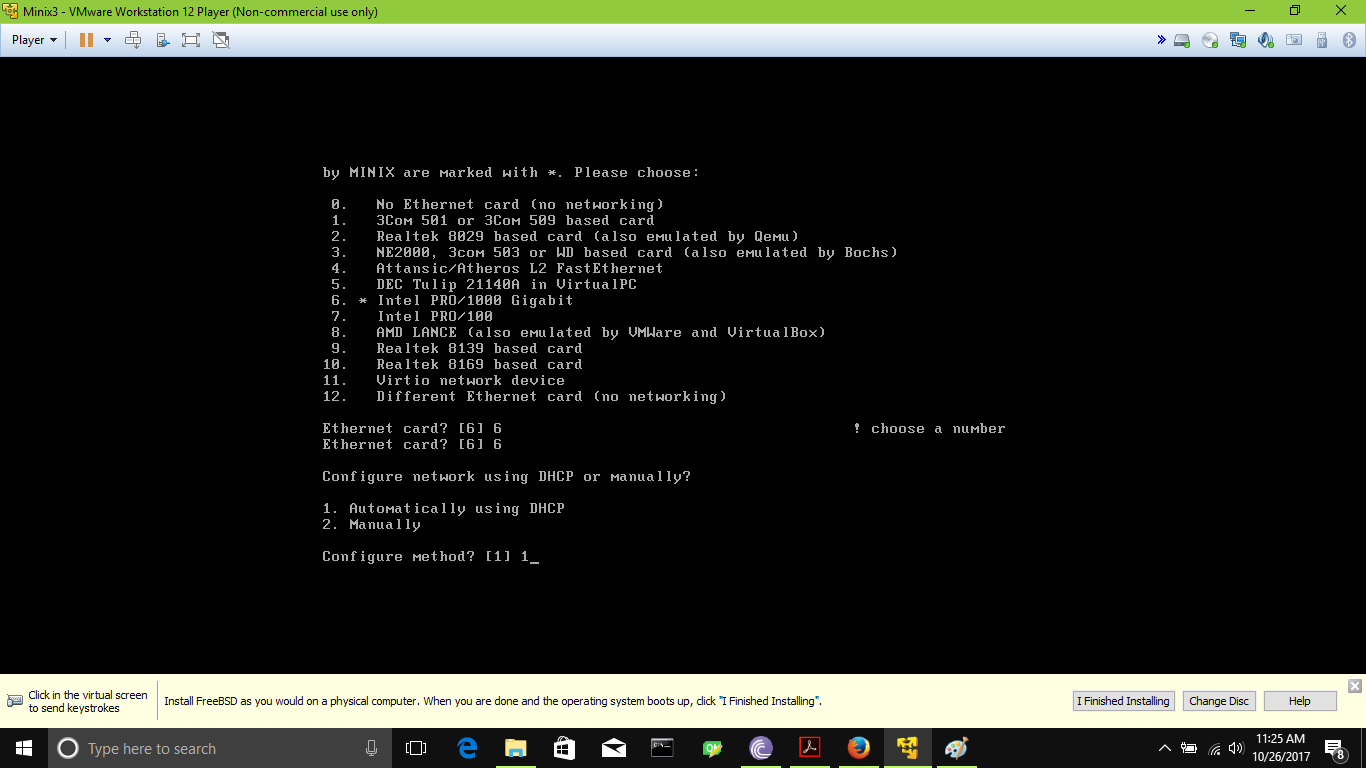


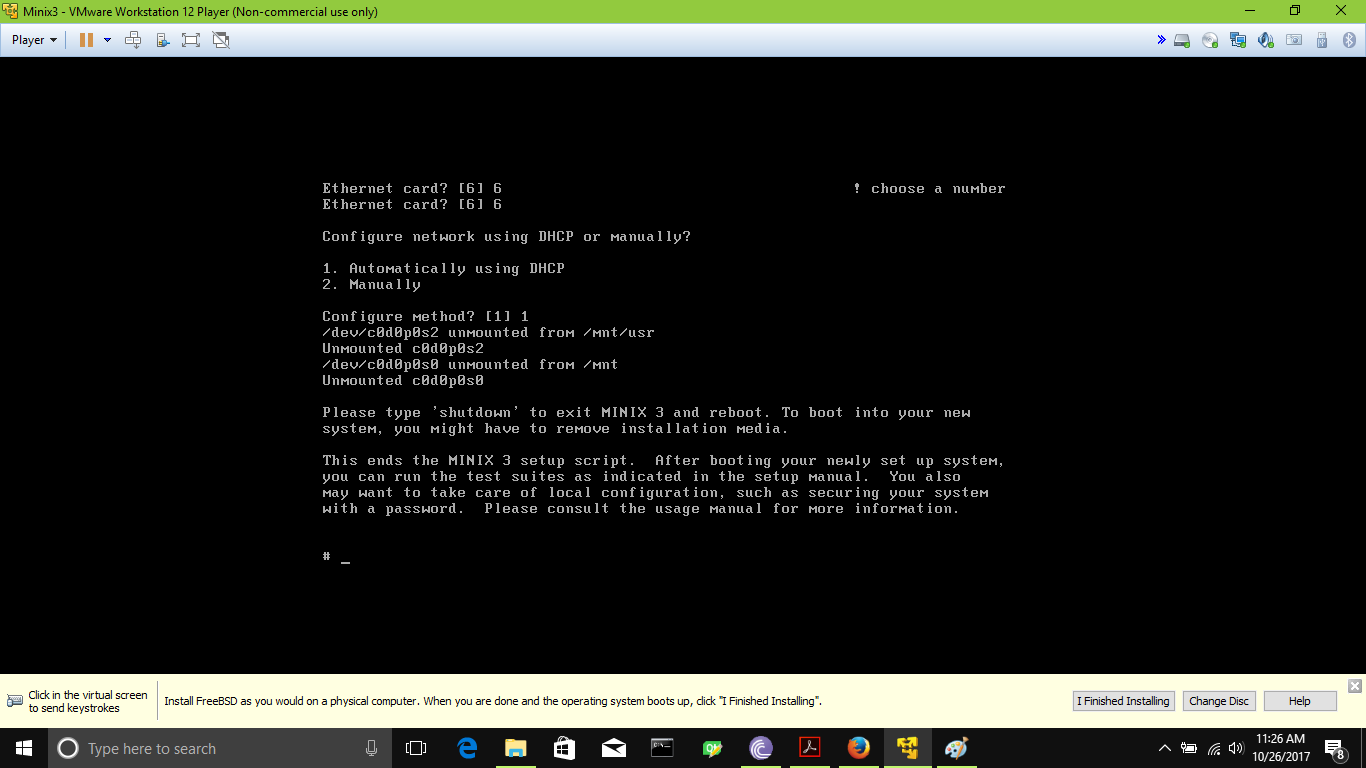




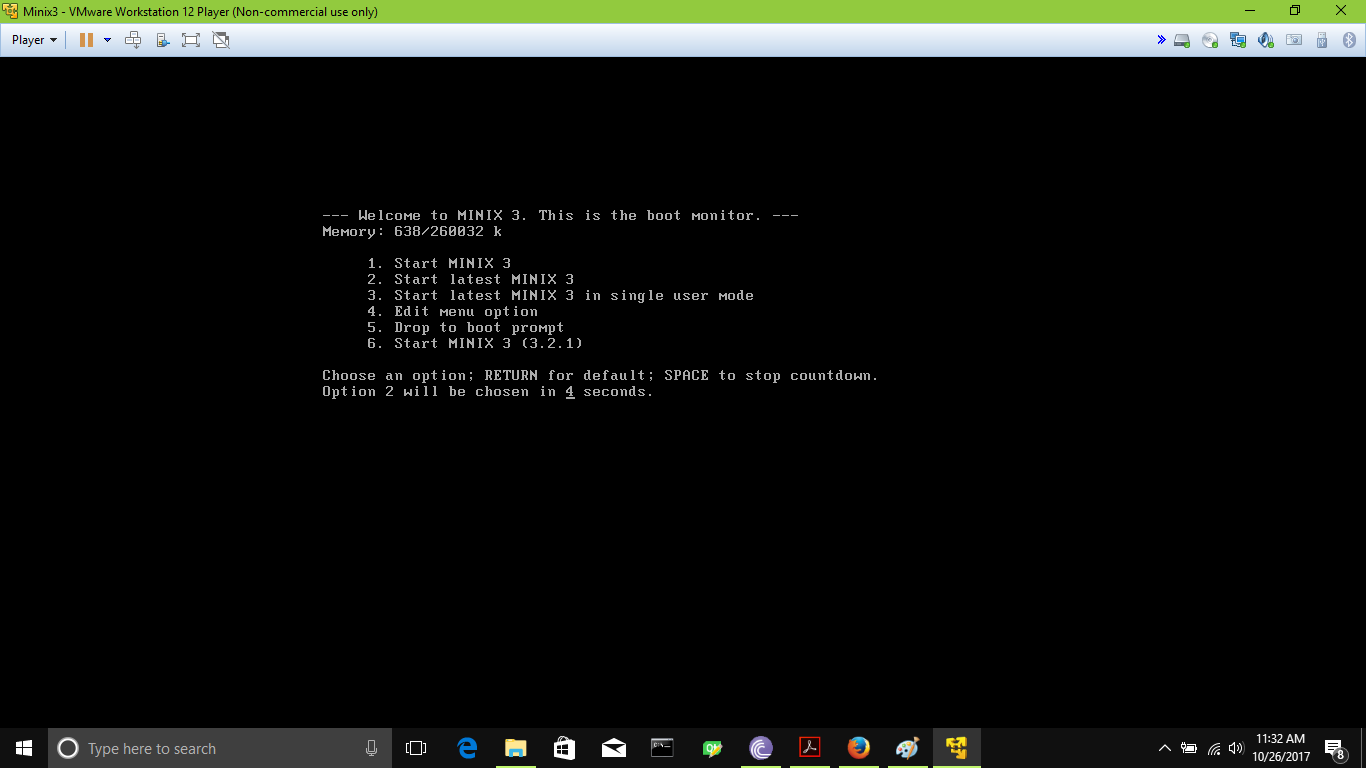




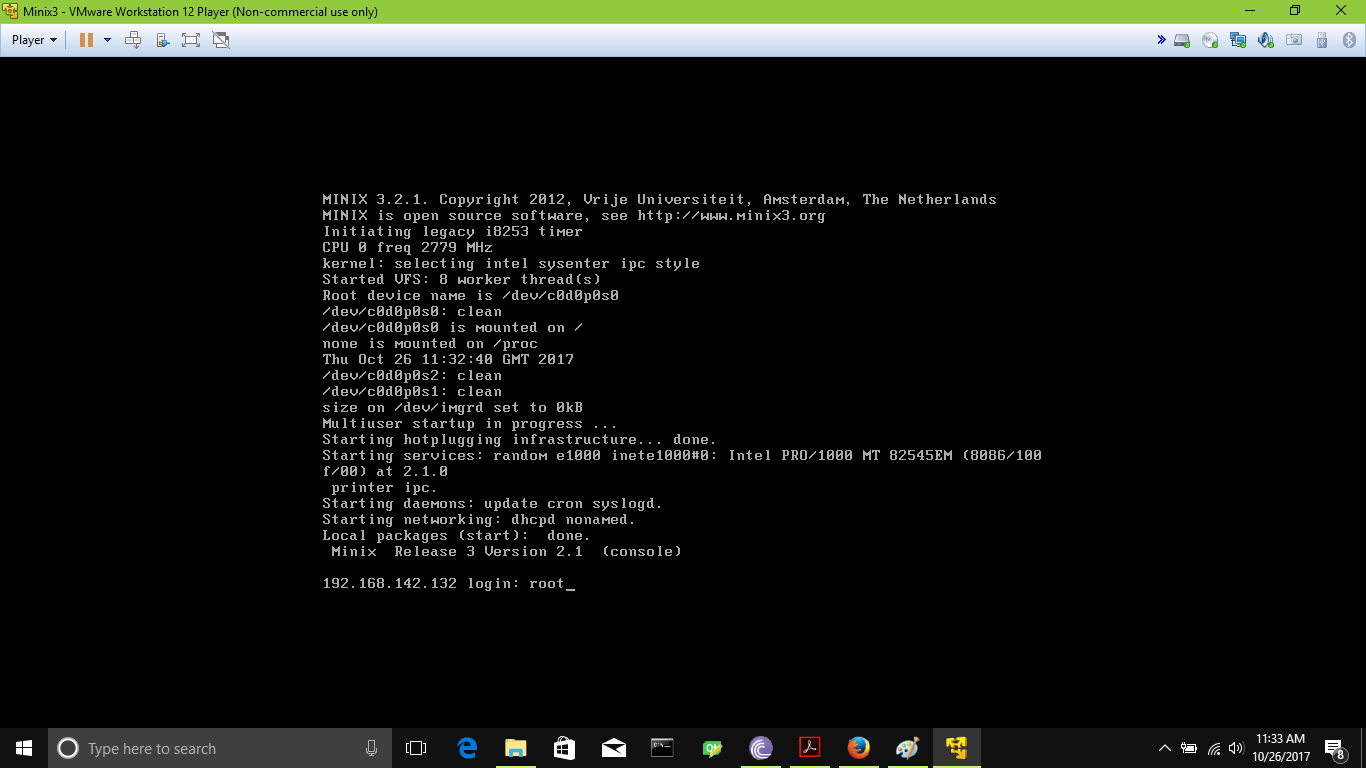




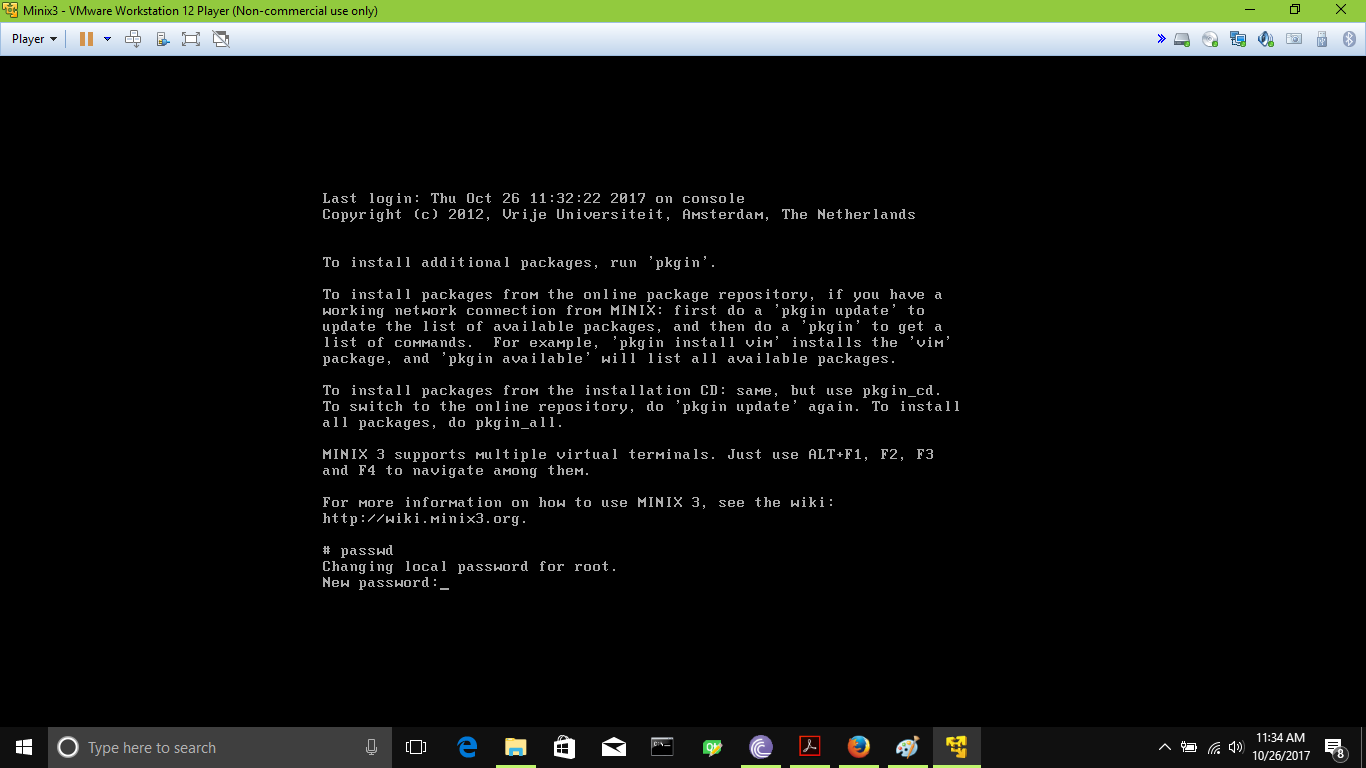
The setup is complete. Reboot the Minix now. Type “Reboot” and press enter. The Minix will restart…



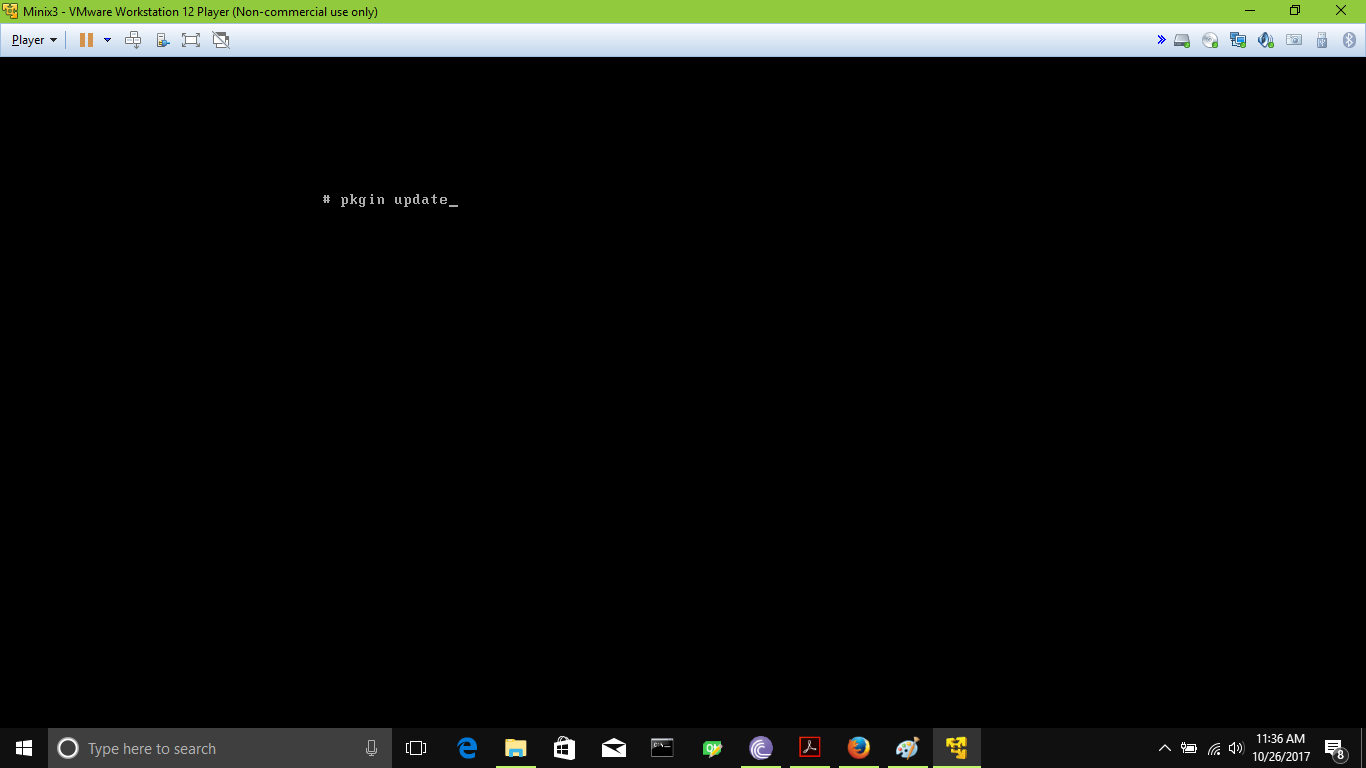
Log in with root.



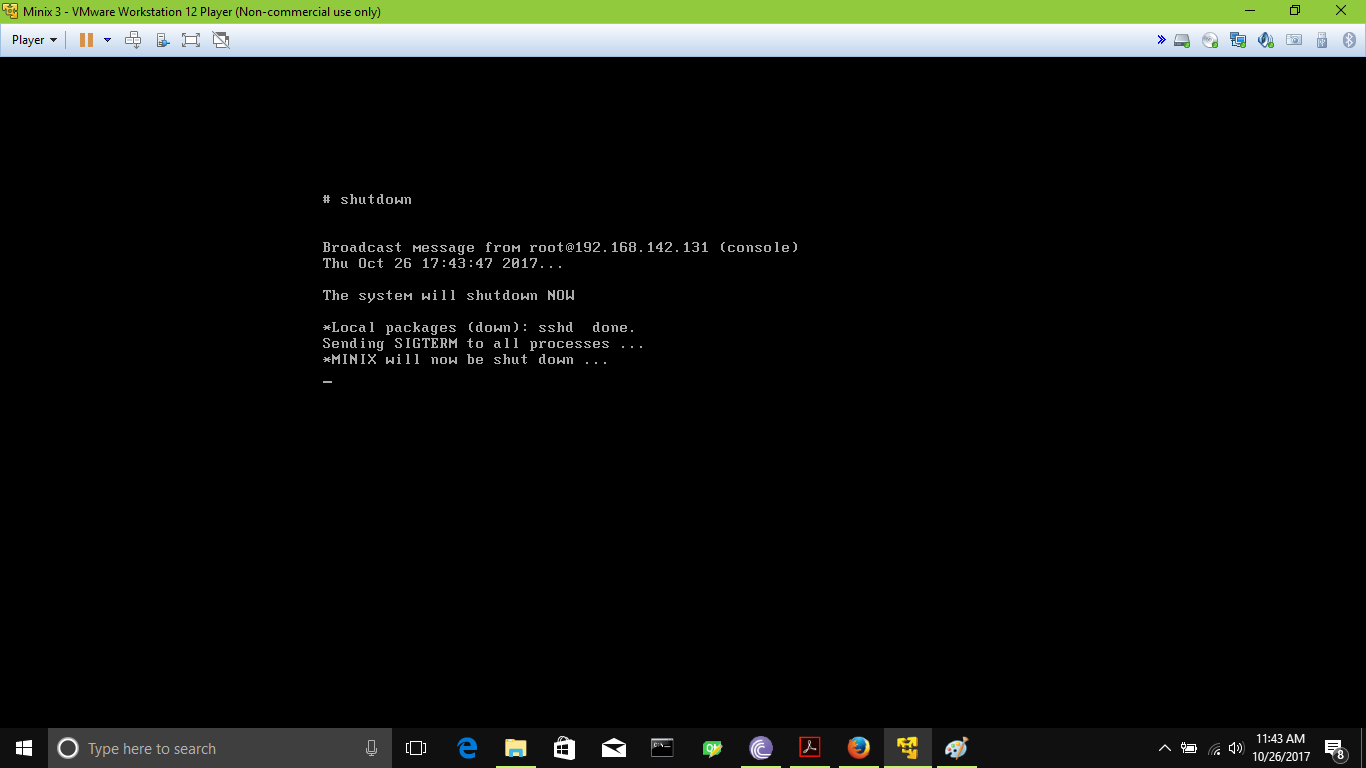
Change your password for root.Type “Passwd” and press enter.

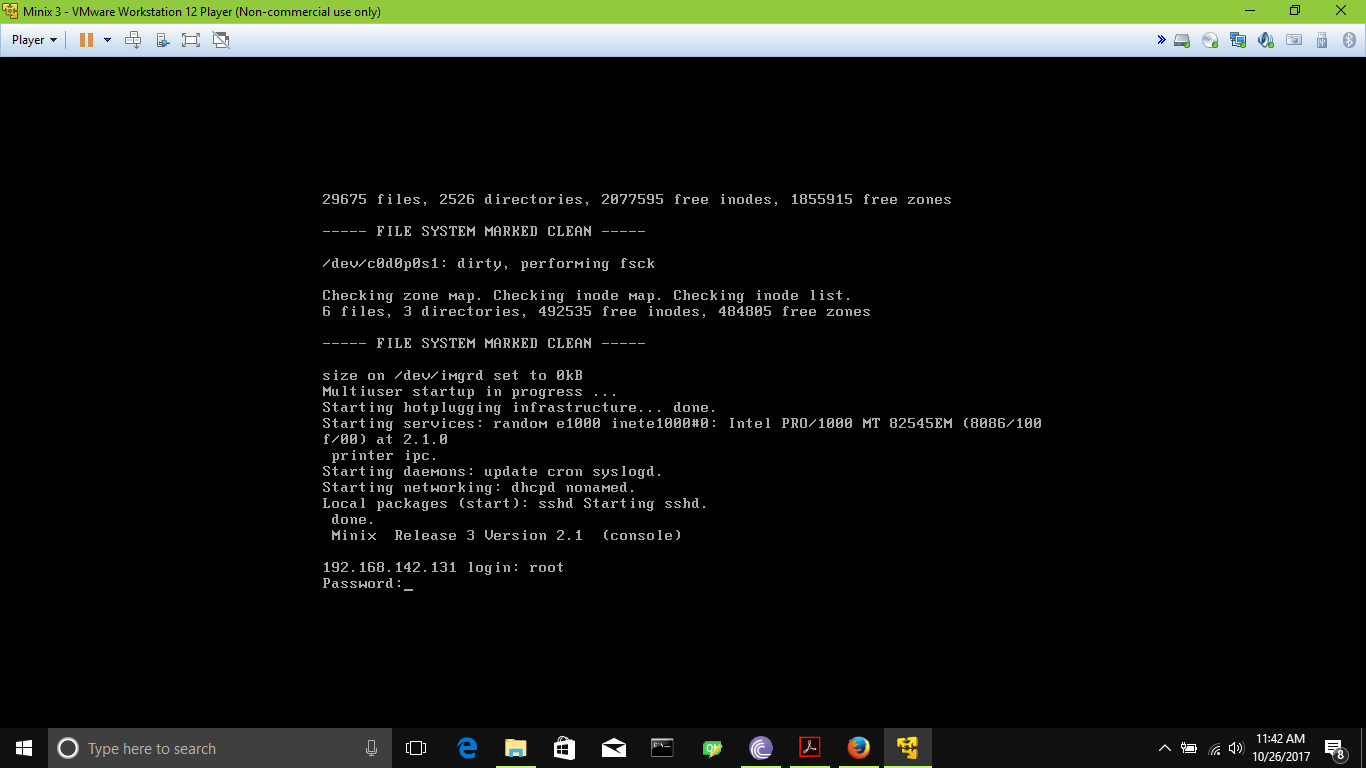


Now set up and update pkgin for install software in Minix. Follow the pictures below and press enter.



Reboot again and Log in with root and enjoy.





**Requirement 2:**

**Part 1:**

Add group…

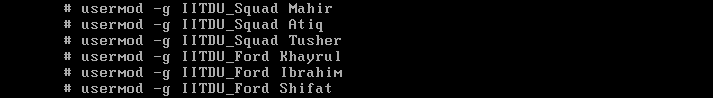
C:\Users\Sand Boa\AppData\Local\Microsoft\Windows\INetCache\Content.Word\AddGroup2.png

Add user in group…The user will be added as Secondary user

C:\Users\Sand Boa\AppData\Local\Microsoft\Windows\INetCache\Content.Word\AddGroup1.png

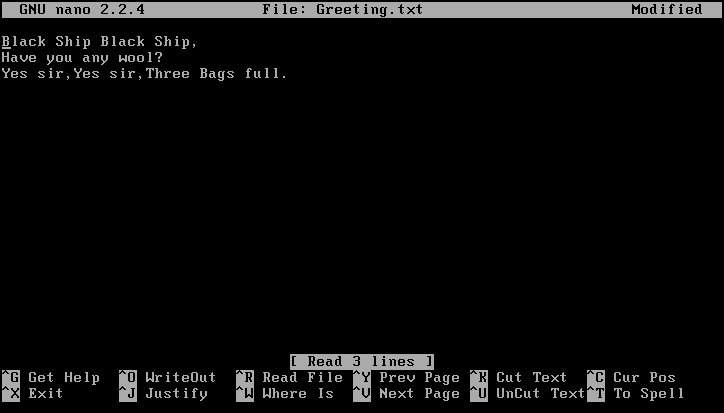
C:\Users\Sand Boa\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Adduser3.png

Now embed it in system. Change user mode to Primary user



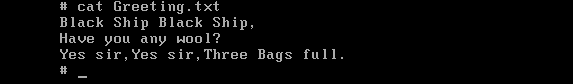
**Part 2:**

C:\Users\Sand Boa\AppData\Local\Microsoft\Windows\INetCache\Content.Word\CreateFile.png



Then press Ctrl+s and ‘y’. to save the file and exit.

Type “cat filename “ to view the file.



**Part 3:**

To change permission there is various method.

The owner of a file or directory can be changed using chmod command, which is stand for “change mood”.The permission variables are u,g,o,a meaning User,Group, Others, All of above and r,w,x.

We also can use bitmasks method with chmod.Each digit is the sum of three bits,with 4 representing read permission,2 representing write permission,1 representing execute permission.

So 755 indicates rwxr-xr-x

C:\Users\Sand Boa\AppData\Local\Microsoft\Windows\INetCache\Content.Word\chmod.png

C:\Users\Sand Boa\AppData\Local\Microsoft\Windows\INetCache\Content.Word\chmodBit.png

[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjh0NinmqDXAhWKLo8KHXXABA8QjRwIBw&url=http%3A%2F%2Fwww.yourownlinux.com%2F2013%2F09%2Fchmod-basics-of-filesdirectories.html&psig=AOvVaw3BmMeAEuvpFDqHiZrud-R6&ust=1509722918394140)

**chmod syntax**

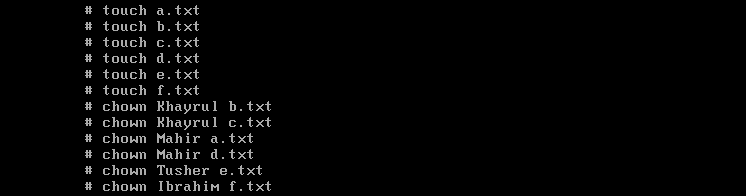
chmod [*OPTION*]... *MODE*[,*MODE*]... *FILE*...

chmod [*OPTION*]... *OCTAL-MODE* *FILE*...

chmod [*OPTION*]... --reference=*RFILE FILE*...

**Part 4:**

Create some file and change owner of the group.

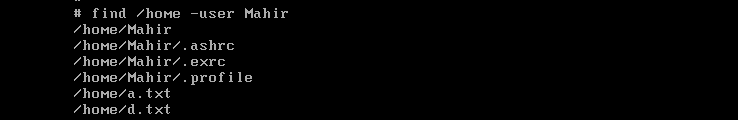


**chown syntax**

chown [*OPTION*]... [*OWNER*][:[*GROUP*]] *FILE*...

chown [*OPTION*]... --reference=*RFILE* *FILE*...

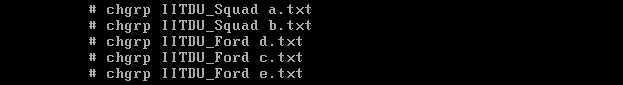
Now query by use name



**find syntax**

find [-H] [-L] [-P] [-D *debugopts*] [-O*level*] [*path*...] [*expression*]

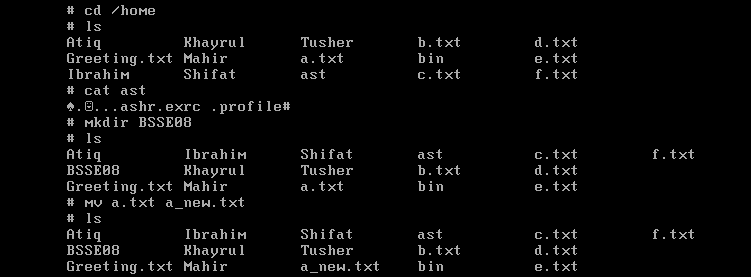
Change the group of files



Now Find file by group.

C:\Users\Sand Boa\AppData\Local\Microsoft\Windows\INetCache\Content.Word\find2.png

**Part 5:**



**Question 2:**

**Requirement 1:**

In Minix 3, the OS do not use sunread and sunerite.Minix 3 do not have mcc compiler.So it’s compiling and execution method is different. It use cc/gcc/mingw as compiler.

In Solaris Minix, mcc first create a compiled file.Next, while running Minix you can type the command.sunread hello > hello to read in the compiled hello program. You can't read it into /usr/bin or /usr/man, since those are read-only file systems. Next, make the program executable with the command chmod +x hello. Then type hello to run the program.

*mcc example.c –o example*

sunread example > example

chmod +x example

example

But in Minix 3,Such imbedding is not allowed.We need to run the object file begin with ./filename.

gcc -o example example.c

./example

To run a c file like a command/application. We need to make a MAKE FILE.

Make the c file and then run the application.

**Question 3:**

**Requirement 1:** Because if they were not, any user would be able to change passwords, or change things dealing with the os, and they would be just as powerful as the root user. With the those commands being setuid programs, ensures that roots are the only ones with high access privileges and that normal users are restricted from certain things.

Go to /usr/src/commands directory to see the source code for commands.

**Requirement 2:**

1.By checking the access flag

2.First It does a validity check, Then it gets the exec file name and sees if the file is executable. Then fetches the stack from the user before the old core image, checks to see if the process text can be shared with that one of already running. Saves file id to allow it to be shared. Patches up stack and copy it from MM to new core image; read in text and data segment. Take care of setuid/setgid bits. Save offset to initial argc,fix mproc fields, tell kernel that exec is done, reset caught sigs.

3.it first checks to see if the mode is correct; then it temporarily opens the file whose access is to be checked; and finally, it checks the permissions.

**Requirement 3:**

A typical setuid program does not need its special access all of the time. It’s a good idea to turn off this access when it isn’t needed, so it can’t possibly give unintended access.

If the system supports the \_POSIX\_SAVED\_IDS feature, you can accomplish this with seteuid. When the game program starts, its real user ID is jdoe, its effective user ID is games, and its saved user ID is also games. The program should record both user ID values once at the beginning, like this:

user\_user\_id = getuid ();

game\_user\_id = geteuid ();

Then it can turn off game file access with

seteuid (user\_user\_id);

and turn it on with

seteuid (game\_user\_id);

Throughout this process, the real user ID remains jdoe and the file user ID remains games, so the program can always set its effective user ID to either one.

On other systems that don’t support file user IDs, you can turn setuid access on and off by using setreuid to swap the real and effective user IDs of the process, as follows:

setreuid (geteuid (), getuid ());

This special case is always allowed—it cannot fail.

Why does this have the effect of toggling the setuid access? Suppose a game program has just started, and its real user ID is jdoe while its effective user ID is games. In this state, the game can write the scores file. If it swaps the two uids, the real becomes games and the effective becomes jdoe; now the program has only jdoe access. Another swap brings games back to the effective user ID and restores access to the scores file.

In order to handle both kinds of systems, test for the saved user ID feature with a preprocessor conditional, like this:

#ifdef \_POSIX\_SAVED\_IDS

seteuid (user\_user\_id);

#else

setreuid (geteuid (), getuid ());

#endif

Here is the source code

#include <sys/cdefs.h>

#include <lib.h>

#include "namespace.h"

#include <string.h>

#include <unistd.h>

int setuid(uid\_t usr)

{

message m;

memset(&m, 0, sizeof(m));

m.m\_lc\_pm\_setuid.uid = usr;

return(\_syscall(PM\_PROC\_NR, PM\_SETUID, &m));

}

int seteuid(uid\_t usr)

{

message m;

memset(&m, 0, sizeof(m));

m.m\_lc\_pm\_setuid.uid = usr;

return(\_syscall(PM\_PROC\_NR, PM\_SETEUID, &m));

}

Parsing the whole source code, We found 71 places where the setuid () was called.

Defined as a function in:

* minix/lib/libc/sys/setuid.c, line 8

Defined as a function prototype in:

* include/unistd.h, line 138

Referenced (in 71 files total) in:

* external/bsd/kyua-testers/dist/run.c:
  + line 360
  + line 361
* external/bsd/libarchive/dist/libarchive/test/test\_write\_disk\_perms.c, line 357
* external/bsd/llvm/dist/clang/lib/StaticAnalyzer/Checkers/CheckSecuritySyntaxOnly.cpp, line 693
* external/bsd/atf/dist/tools/user.cpp, line 59
* external/public-domain/xz/dist/src/xz/file\_io.c, line 584
* bin/ksh/misc.c, line 330
* bin/sh/main.c, line 192
* include/unistd.h, line 138
* tests/kernel/kqueue/t\_proc2.c:
  + line 69
  + line 70
* tests/fs/vfs/t\_unpriv.c:
  + line 57
  + line 69
  + line 96
  + line 110
  + line 146
  + line 162
  + line 206
* tests/fs/nfs/nfsservice/rpcbind/rpcbind.c:
  + line 207
  + line 208
* tests/lib/libc/sys/t\_msgsnd.c, line 291
* tests/lib/libc/sys/t\_revoke.c, line 148
* tests/lib/libc/sys/t\_chroot.c, line 283
* tests/lib/libc/sys/t\_issetugid.c, line 135
* tests/lib/libc/sys/t\_msgctl.c, line 134
* tests/lib/libc/sys/t\_setuid.c:
  + line 45
  + line 53
  + line 60
  + line 67
  + line 76
  + line 94
* tests/lib/libc/sys/t\_kill.c:
  + line 171
  + line 184
* tests/lib/libc/sys/t\_mkdir.c, line 130
* usr.sbin/syslogd/syslogd.c, line 543
* usr.sbin/chroot/chroot.c:
  + line 183
  + line 184
* usr.sbin/traceroute/traceroute.c:
  + line 919
  + line 952
* usr.bin/lock/lock.c:
  + line 162
  + line 163
* usr.bin/login/login.c:
  + line 599
  + line 601
* usr.bin/newgrp/newgrp.c:
  + line 133
  + line 134
* usr.bin/calendar/calendar.c, line 635
* usr.bin/whereis/whereis.c, line 74
* usr.bin/su/su.c:
  + line 351
  + line 352
* usr.bin/rsh/rsh.c:
  + line 207
  + line 264
* crypto/external/bsd/heimdal/dist/kdc/main.c:
  + line 94
  + line 95
* sbin/ping/ping\_hostops.c, line 52
* sbin/ping/prog\_ops.h, line 76
* sbin/ping/ping.c, line 274
* sbin/shutdown/shutdown.c, line 523
* lib/libutil/passwd.c, line 266
* lib/libutil/login\_cap.c:
  + line 707
  + line 708
* lib/librmt/rmtlib.c, line 347
* lib/libc/net/rcmd.c:
  + line 448
  + line 449
* minix/commands/arp/arp.c, line 180
* minix/commands/term/term.c, line 369
* minix/commands/rlogin/rlogin.c, line 370
* minix/commands/hostaddr/hostaddr.c, line 180
* minix/commands/ifconfig/ifconfig.c, line 112
* minix/commands/cron/cron.c, line 185
* minix/commands/mail/mail.c:
  + line 119
  + line 195
  + line 618
  + line 720
* minix/servers/rs/manager.c:
  + line 655
  + line 817
* minix/fs/pfs/pfs.c, line 398
* minix/tests/test56.c:
  + line 1113
  + line 1146
* minix/tests/test25.c, line 614
* minix/tests/test87.c, line 46
* minix/tests/test33.c:
  + line 49
  + line 54
  + line 84
  + line 108
  + line 132
  + line 165
  + line 189
* minix/tests/test65.c, line 86
* minix/tests/test60.c:
  + line 79
  + line 80
  + line 85
  + line 86
  + line 91
  + line 92
  + line 224
  + line 240
  + line 248
* minix/tests/test5.c, line 254
* minix/tests/test85.c, line 496
* minix/tests/test17.c, line 665
* minix/tests/test46.c:
  + line 40
  + line 49
* minix/tests/test89.c:
  + line 288
  + line 389
  + line 413
  + line 585
  + line 878
  + line 969
* minix/tests/test42.c, line 1132
* minix/tests/test88.c:
  + line 109
  + line 2954
* minix/tests/test73.c, line 48
* minix/tests/test34.c:
  + line 216
  + line 489
* minix/tests/test35.c, line 226
* minix/tests/test11.c:
  + line 43
  + line 47
  + line 95
* minix/tests/test79.c, line 298
* minix/tests/test18.c, line 928
* minix/usr.bin/trace/service/pm.c, line 1287
* minix/net/inet/inet.c, line 242
* minix/lib/libc/sys/setuid.c, line 8
* libexec/rshd/rshd.c, line 705
* libexec/ftpd/ftpd.c, line 1627

Source code has been modified in corresponding positions and Make file is made to compile the source code and then installed later.