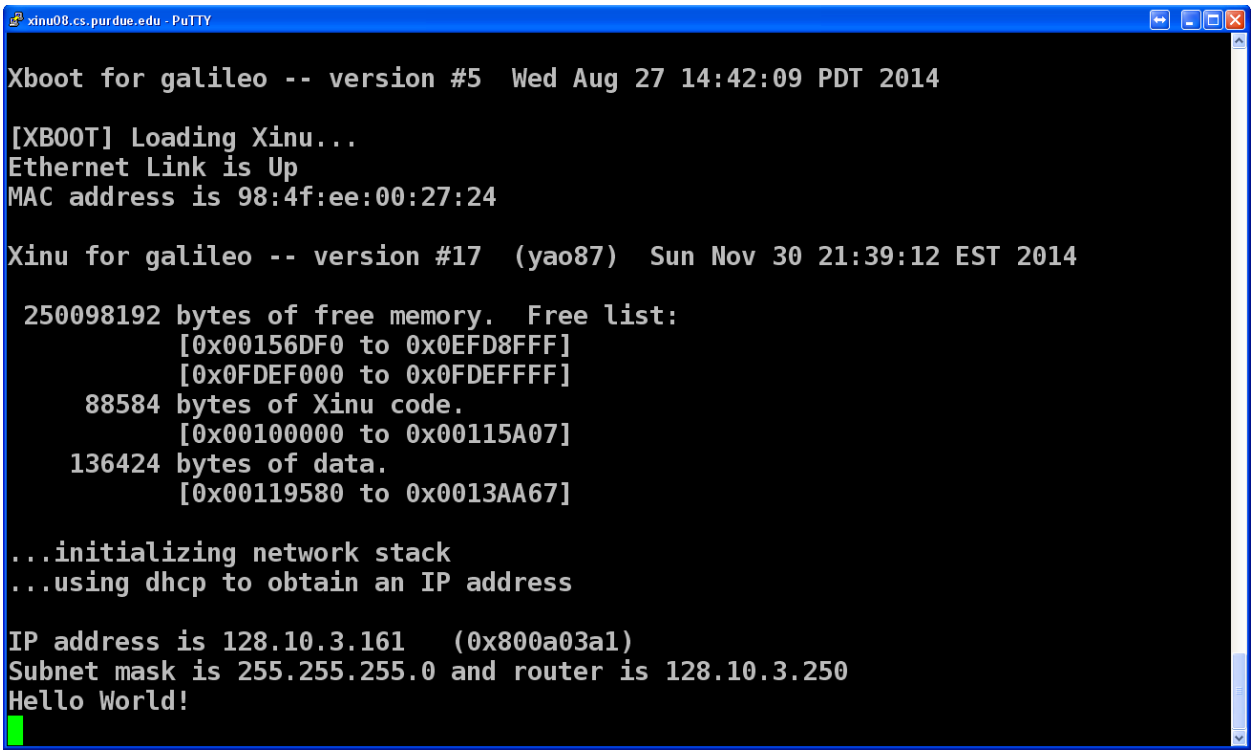
There are 5 switches for tests in main.c:

1. Shell test
2. Helloworld test
3. Hash table test
4. Load_library test
5. Load_library test for too many functions and duplicate functions

Test Result:

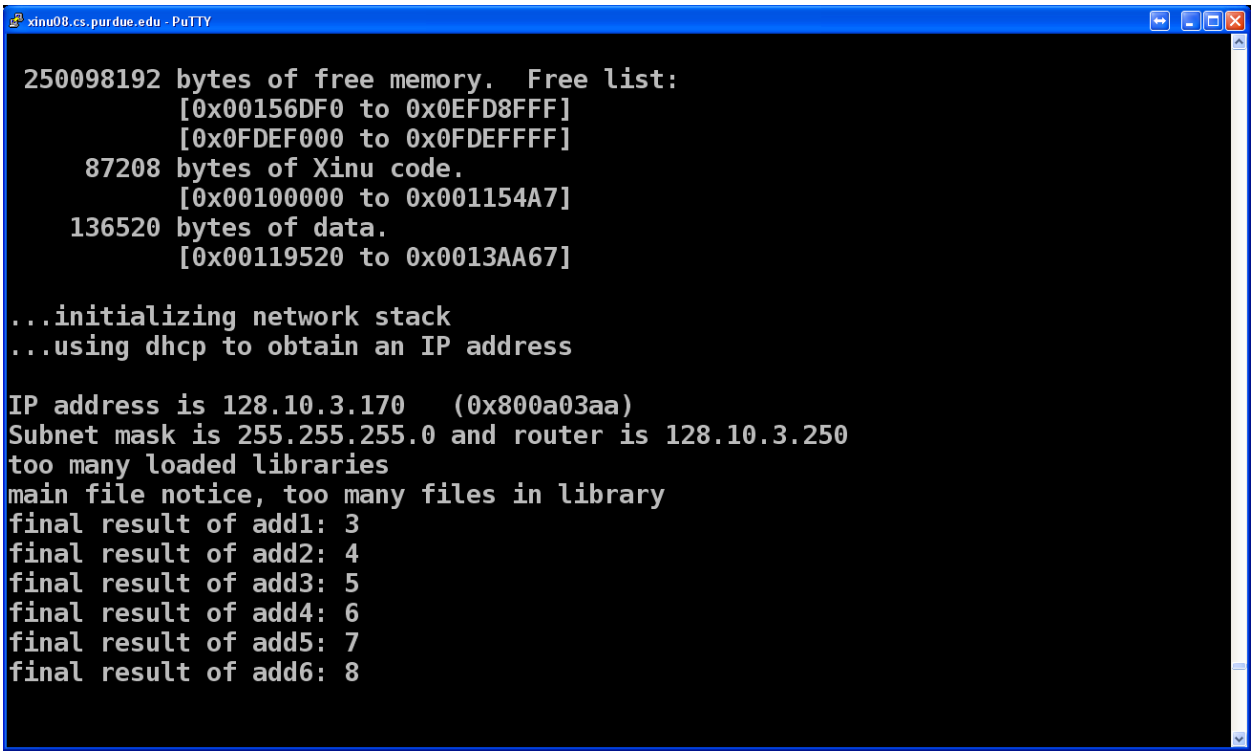
1. Helloworld



2. loadlibrary with initial value 2
   For load_library, I have initialized a hash table in the xinu system and add the new file functions into the hash table. If I need the functions, I just get from the hash table.
   I have several files to load into the library:
   myadd.c myadd1.c … myadd6.c

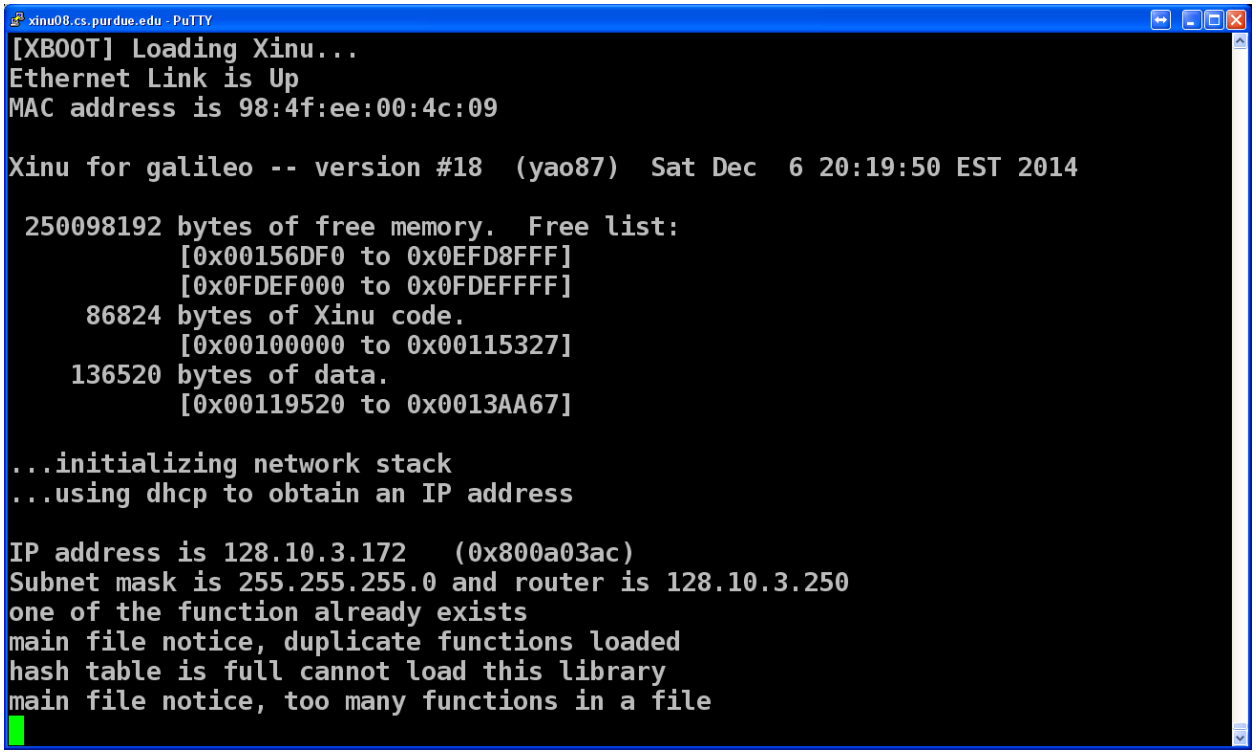
myadd.c, myadd1.c myadd2.c, myadd3.c implements the functions: add1, add2, … , add8. Every file has two functions inside.
When I tried to load myadd3.c which is the fourth file, the load_library return syserr and prints the main file notice in main.c.

The function works well while loaded from the libraries.



Myadd5.c file has duplicate add1 which has already been loaded by myadd.c so there would return an error.

Myadd6.c file has so many functions which is more than 10, so there return an error.



3. Shell implementation

   Ls:

   Filter out the “..” and “.” when print out, but can still use ls ‘.’ to list the self directory.

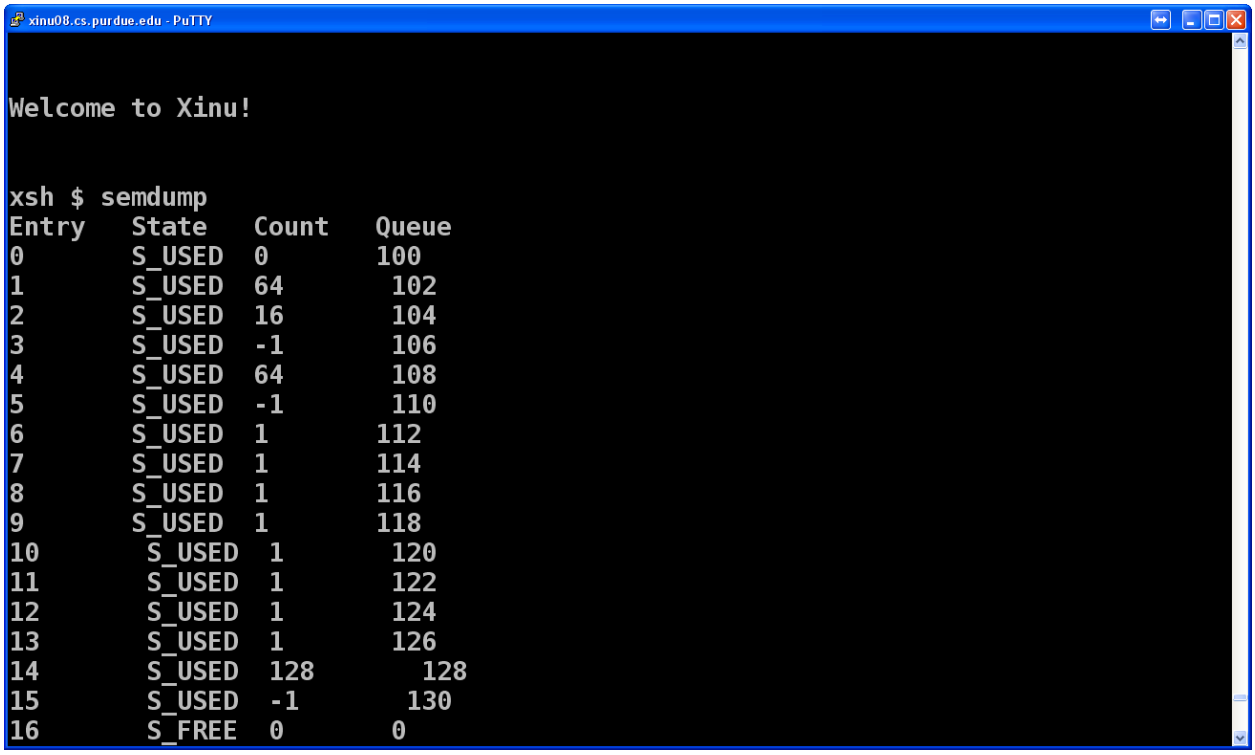
```
xinu08.cs.purdue.edu - PuTTY
    \ \/ /     | |    |  \| |  | |  | |
    / /\ \    _| |_   | \   |  | |  | |
   / /  \ \  |     |  | | \ |  \  --  /
   --    --   -----    -   -    ----
-----------------------------------------


Welcome to Xinu!


xsh $ ls
helloworld.c
Makefile
ls
ls.c
semdump.c
testdir/
helloworld
xinu.elf
#helloworld#
myadd
myadd.c
semdump
rfserver
```

Ls directory and ls to a file which returns panic, ls '..' is not working, but ls '.' works well. ls testdir(created directory which has testdir.c) it will return the testdir.c file which is in testdir.

```
xinu08.cs.purdue.edu - PuTTY
xsh $ ls testdir
testdir.c
xsh $ ls myadd
myadd is a file, cannot be read
xsh $ ls .
helloworld.c
Makefile
ls
ls.c
semdump.c
testdir/
helloworld
xinu.elf
#helloworld#
myadd
myadd.c
semdump
rfserver
xsh $ ls ..
No Such Directory or File
```

Semdump:

For the semdump, I have implemented a function in xinu/system which return the semtab, since semtab is not in the xinu.elf file, it's necessary to implement this file and relocate to find the semaphore table.



The states after 16 are all free.

- The details behind your implementation. As part of this discussion write answers to the following questions:
    - The separation between a Xinu ELF file and the running image leads to a potential problem: if the ELF file is changed (Xinu sources are recompiled) after an image starts to run, symbol table addresses in the ELF file may no longer match the locations of items in the running image. How can you ensure the ELF file read at run time matches the image that is executing?

      The xinu.elf file has been loaded into the memory which has already been fixed during running process. So if the elf file has changed, the memory won't change, since there is no read operation again in the code. Try to avoid the elf file change, my way of doing this is trying to read this elf as soon as possible in load program code in order to avoid the change of elf file.

    - What is the most difficult aspect of the project? Why?

      The most difficult aspect of the project is reading the elf file manuals.

The specifications and details in the manual is not enough for understanding the procedures of implementing the relocation of functions if you never did that before. Thanks for TA's help, I finally figure out what S P A stands for in the relocation equation. This is really interesting project and rfserver gives me a brief understanding of remote file system. There would be sometimes unknown bugs, (maybe due to the file operation bugs), sometimes cannot connect to the server so cannot open the file, but the whole process of doing this is full of fun.