README.md

School of Engineering and Applied Science (SEAS), Ahmedabad University

Operating Systems Lab - CSE 341

Faculty: *Prof. Mansukh Savaliya*

Project Progress Report : *Week 1*

Group Number: *3*

Group Members:

Name	Enrolment Number
Yashil Depani	AU1841005
Vidish Joshi	AU1841019
Manav Patel	AU1841037

OSProject - toaruOS

An operating systems project on understanding and solving issues for the open source OS - toaruOS.

This repository and read me contains the work done, issues faced and implementation done by us on the original OS. These steps will be similar in some cases to that of the original OS and different in some others.

Index

Installation guide

Guide to install ToaruOS in Ubuntu.

First and foremost, ***QEMU*** and ***GCC*** are required to be already present in the system and it is assumed that they are already present in the system.

It is also required that ***Git*** version control system is also installed in the system. If not already installed, it can be installed using the command,

```
sudo apt install git
```

make is also required to be installed in the system for installing this OS. Use the below command if not already present.

```
sudo apt install make
```

Next clone the ***toaruOS*** GitHub repository in your desired folder.

```
git clone https://github.com/klange/toaruos
```

xorisso, ***autoconf***, ***gnu-efi***, ***GMP***, ***MPFR*** and ***MPC*** utilities are required to install the OS. If not present, you can install them using these commands;

Install ***xorisso***:

```
sudo apt-get update -y
sudo apt-get install -y xorriso
```

Install ***autoconf***:

```
sudo apt-get update -y
sudo apt-get install -y autoconf
```

Install ***gnu-efi***:

```
sudo apt-get update -y
sudo apt-get install -y gnu-efi
```

Install ***GMP***:

```
sudo apt-get update -y
sudo apt-get install libgmp3-dev
```

Install ***MPFR header***:

```
sudo apt-get update -y
sudo apt-get install libmpfr-dev
```

Install ***MPC header***:

```
sudo apt-get update -y
sudo apt-get install libmpc-dev
```

After installing these dependencies, we are all set for installing the OS.

Go inside the `toaruOS` folder and run the make command.

```
cd toaruOS
make
```

Say *yes* when asked to build the toolchains and wait till the system is installed.

After successful implementation of these commands, an image ISO file name `image.iso` will be generated inside the folder. We can run the OS through this file.

Now, we can run the OS in QEMU. For the first time, we need to provide memory space to the OS image file to allow it to function.

Run the OS for the first time using this command:

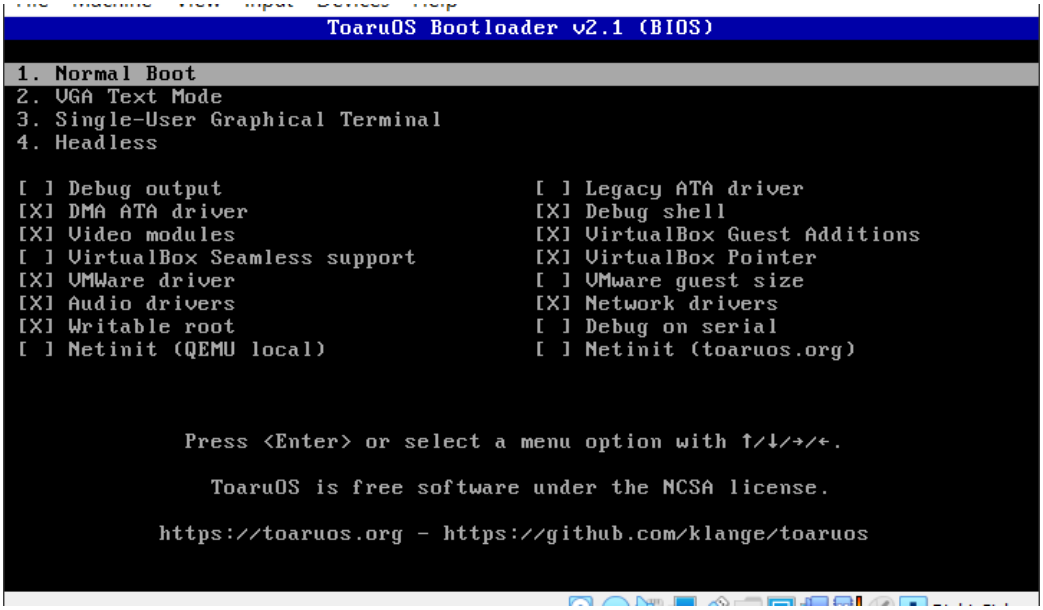
```
qemu-system-x86_64 -boot d -cdrom image.iso -m 1024
```

Here we are providing 1 GB memory to the OS. It is recommended more than enough.

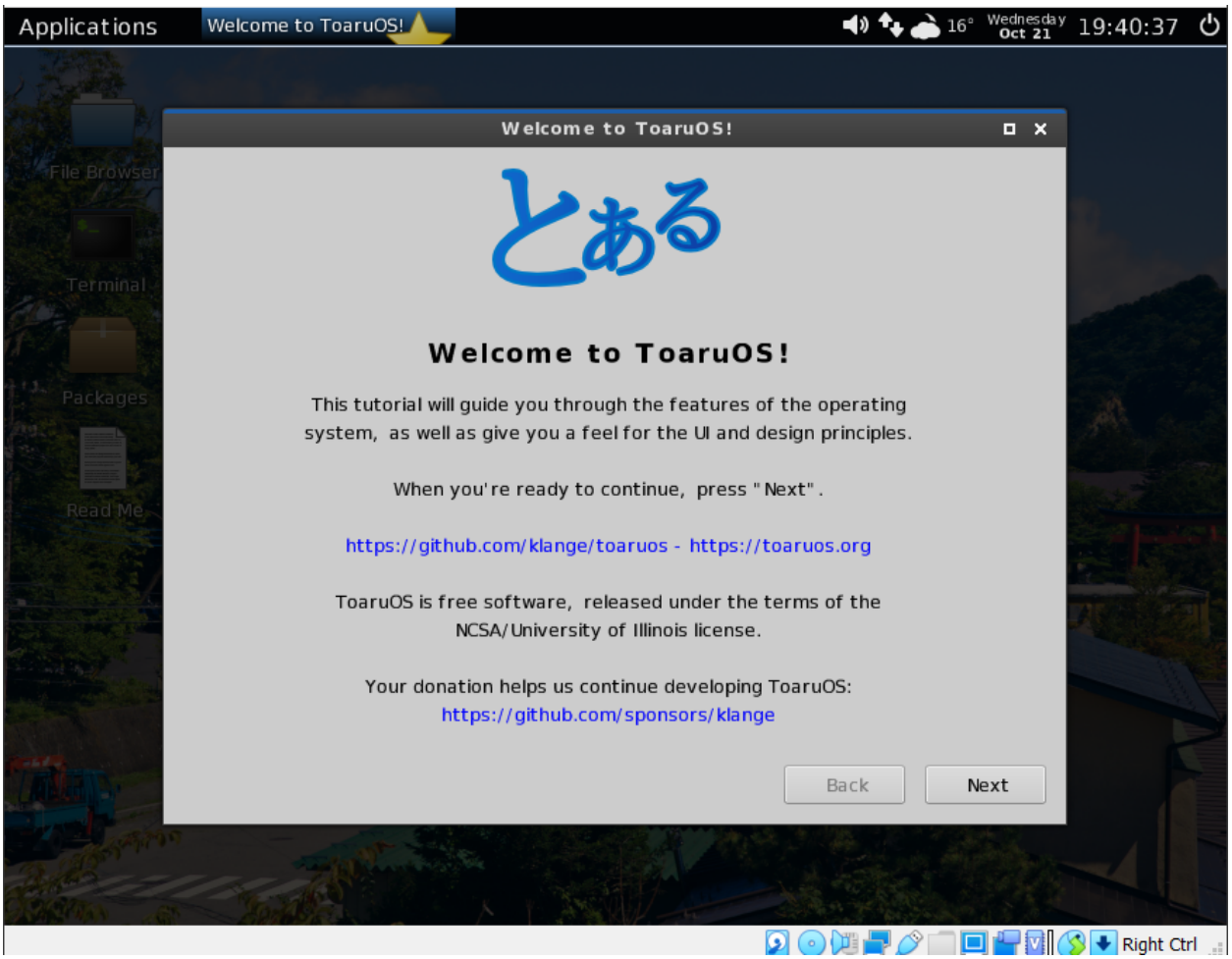
From the following times, we can run it through the command:

```
qemu-system-x86_64 -boot d -cdrom image.iso
```

Upon successful implementation, you will get these screen.

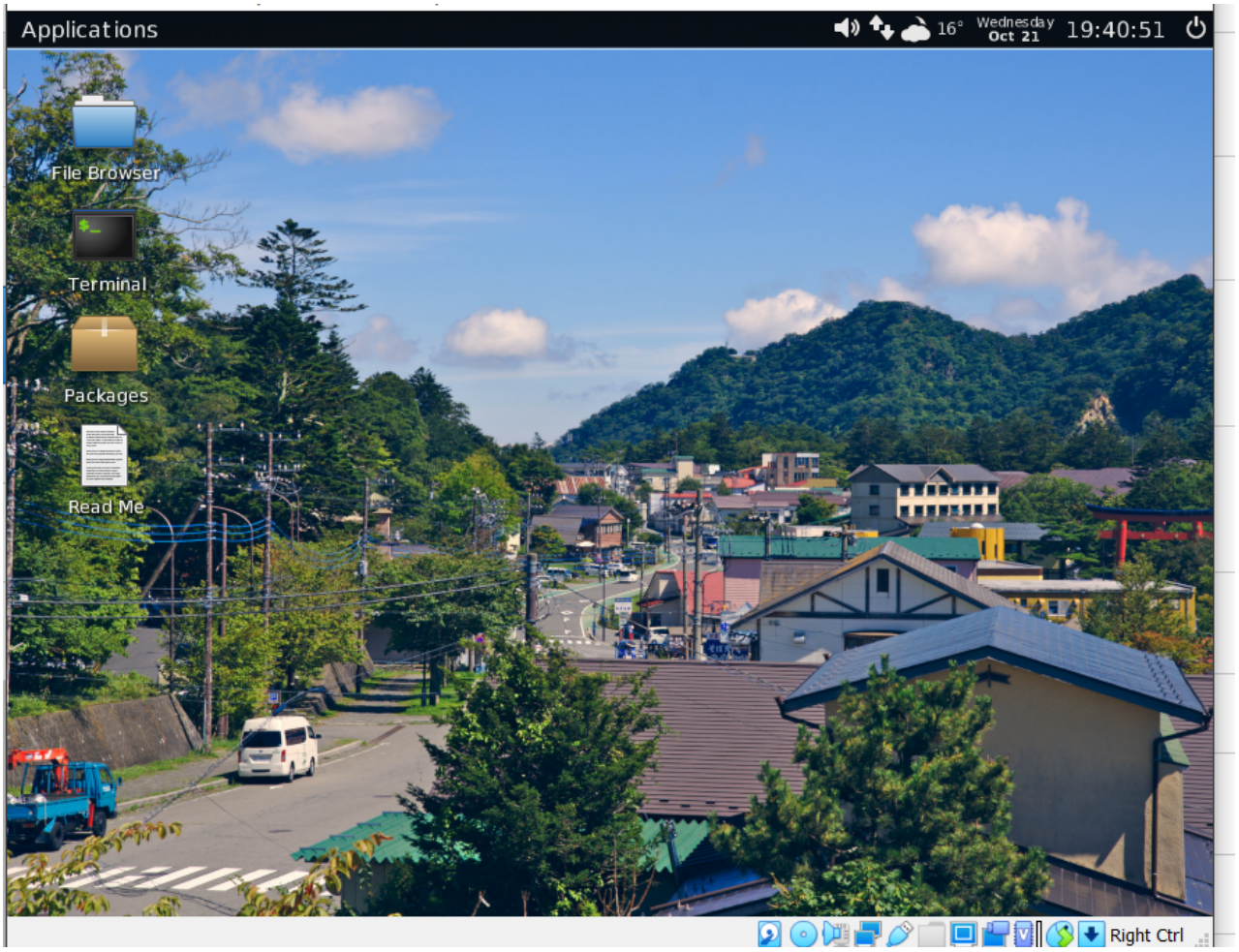


You can select `Normal Boot` and get this screen.



You will be taken across basic tutorials for how to work around the OS. Go through them and press `NEXT` to finish them.

The desktop contains shortcuts to `terminal`, `File Browser`, and `Readme` file.



We have successfully installed and run the OS!

Understanding and Resolving Issues

FAT Driver Issues

ToaruOS has a number of running issues, the most significant of which is File System issues. There are various issues currently open regarding File Systems in it's GitHub repository such as this , this and this.

In the initial stage, we have decided to solve this issue of File System drivers through FAT file system. For this, we first get an understanding of FAT file system.

Understanding FAT32 Drivers

We watched the following YouTube videos to understand FAT32 Drivers.

```
* https://www.youtube.com/watch?v=_h30HBYxtws
* https://www.youtube.com/watch?v=HjVktRd35G8
```

File Systems are the standards for organizing data on storage devices and they are applied when one formats a drive or partition.

The choice of these drivers is based on the type of OS one is working on and the type of drive one is formatting. *For example*, in windows if we want to format the drive, we can chose NTFS file system of exFAT. Whereas in Linux, if we want to format a drive, we can chose from FAT32, NTFS or EXT4 drivers.

Differences between File Systems:

FAT12, FAT16 or FAT32:

FS divide storage space of the drive into virtual compartments or *clusters* and maintain an *index* of where individual files are located and available free space. The 1st DOS windows file system was known as *File Allocation Table* = "*FAT*"

	Max File Size	Max Volume Size
FAT12	32 MB (8 KB clusters)	32 MB (8 KB cluster)
FAT16	2GB / 4GB	16GB (256 KB cluster)
FAT32	4 GB	32 GB (Windows format), 2 TB other OS, 16 TB theoretical

FAT32 is very popular due to its wide compatibility with various OS and is widely used for formatting removable media.

NTFS

Currently, the most popular windows file system is the *New Technology File System - NTFS*. File size limit of 16 exabytes. NTFS is a journalling file system. NTFS also allows file permissions unlike FAT32. Limited OS compatibility.

exFAT

Extended File Alliocation Table - exFAT

File system optimized for high capacity USB flash drives and memory cards. Maximum File Size = 16 exabytes. Wider different OS support than NTFS.

ext2, ext3, ext4

Extended File System (ext)) was launched for linux. ext3 introduced journalling. Maximum file size upto 16TB and maximum volume size of upto 1 EB. NO OS except Linux supported.

Choosing a FS:

For a system drive, chose on the basis of OS. NTFS for windows, ext4 for Linux, etc.

For USB drives and removable drives under 32 GB, use FAT32 for cross-platform support.

exFAT for removable drives for capacity greater than 32 GB capacity or for storing files greater than 4 GB.

FAT File System

Clusters on a disk are used to hold file content and FS structures.

On a FAT file system there are 2 type of clusters:

- Data clusters containing the file contents
- Directory clusters containing directory entry structures.

Directory Clusters:

They hold the FS metadata for all the files.

It contains file names, timestamps and starting cluster for files.

A cluster used for directory entries will not be used for file data.

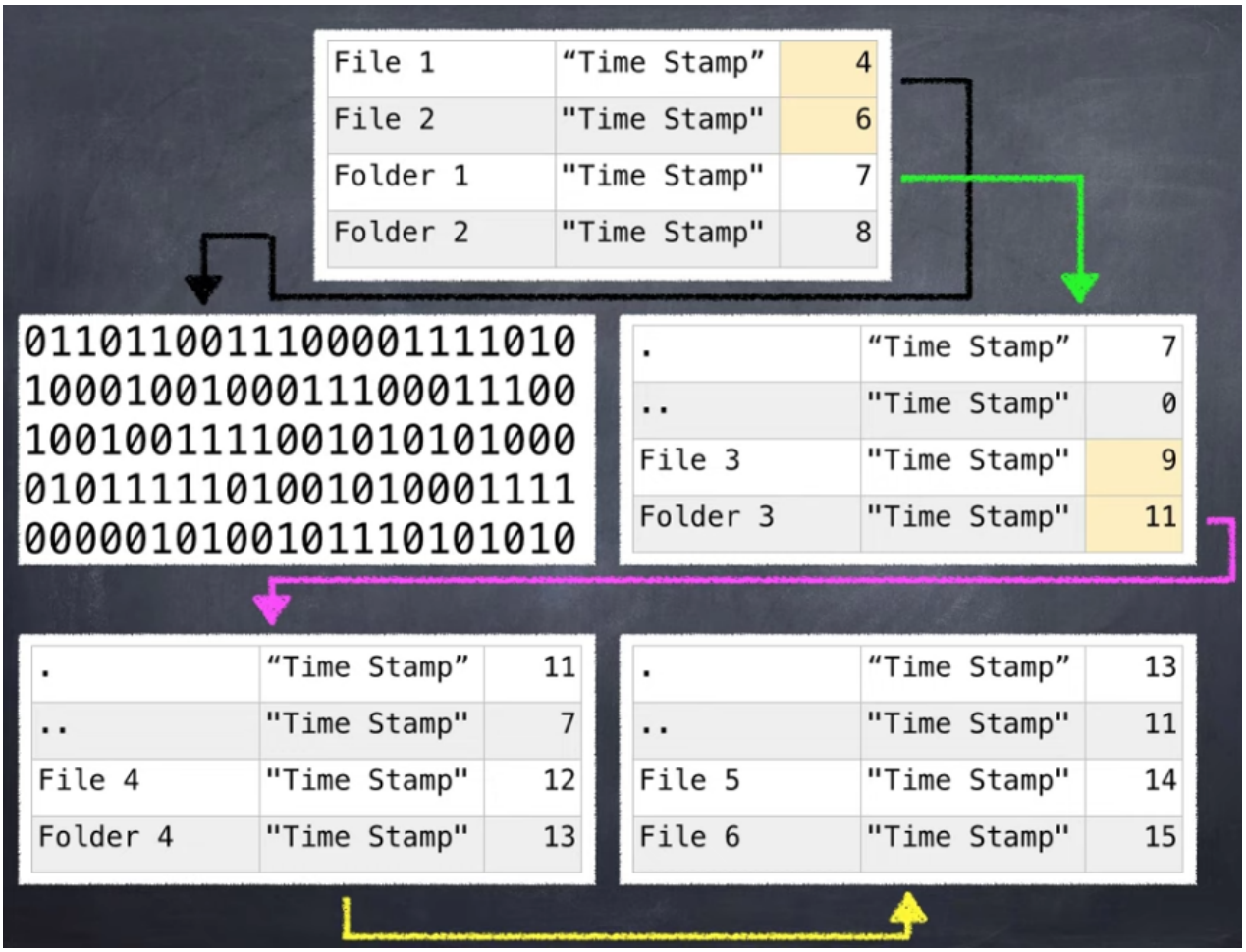
Root Directory:

The very beginning of the disk contains Root Directory.

It contains first set of directory entries. It is in the first set of clusters.

STRUCTURE:

The structure of Root Directory contains the File Names, with their Time-Stamps and their starting cluster address.



Here, the " **.** " file inside Folder 1 has cluster address same as that of Folder 1. This is for the system to know where these files are located.

The " **. .** " files contains the cluster address of the parent directory.

And with the same logic we can go inside Folder 3 and then Folder 4, etc.

Here, we may notice that there are no references to cluster 5 and 10 in all these cluster addresses. Most likely, these clusters have been used by File 1 which uses cluster 4 and 5 File 3 which uses cluster 9 and 10. But we cannot know that for a fact. Because, the directory entries only contain the starting point of the clusters.

The detailed mapping of all the clusters is available in the FAT table.

FAT table is located at the beginning of the disk and it contains:

- File to cluster mapping. This map maps out all the cluster in the system with respective files cell by cell. Each cell in this table represents a cluster. The addressing is dependent on the FAT version.

FAT Cell Possibilities

- 0 - unallocated
- `FF7`, `FFF7` or `FFF FFF7` (12, 16, 32) - bad cluster
- `FF8`, `FFF8`, `FFF FF8` (12, 16, 32) - end of file
- All other values mean the cluster is allocated

Plan for next week

This is the tentative plan for the upcoming week.

We want to collect and understand resources that help us to understand the working structure of FAT drivers and start coding if possible. These are some of the links that we intend to start going through:

1. Documentation - Link

2. Documentation 2 - Link2

For coding, we will start understanding the structures and header files built and used by the authors. As they are written from scratch by the current authors, the structures are somewhat different than the conventional standard header files. We also will be looking for resources to start coding.

Contribution

Installation of the OS - All 3 members

Understanding file systems, collecting resources and explaining concepts to each other. - All 3 members

Making the report - All 3 members