# Shell Design

Luiz Medeiros

03/31/2015

Below is the design to be used on the implementation of the Shell program. This program will serve as the graphical interface between the kernel, the user and the system.

## Overview

The shell will be a separate program loaded after the kernel and essential functionalities of the system have already been loaded into memory deemed successful. Thus, it is expected that after the system is done loading initial and essential resources, the kernel will call the shell. It will then execute through a separate, sub-process or a second thread.

In addition, the Shell should perform the following functions:

Create Directory

Change Directory

Copy File From To

Display System Date and Time

Display Contents of File

Delete File

Display List of Files in Directory

Quit

These have been predefined as follows:

mkdir "dirName"; // makes directory;

cd " directory/name" // changes directory

cd .. // come back one directory above

copy "original/File/location" "file/new/location" // syntax for copying.

date // to display date and time

cat "filename" // to display the contents of a specific file

delete "filename" // to delete a specific file

delete /a // to delete all

dir // show what is in current directory

dir "path/name"

quit // quits from the current running instance of the shell;

## Design Phase:

It follows that the system will follow the flow given below:



Figure , Flow chart of the PC turning on, demonstrating where the Shell comes in

The user will only see the display of the shell booting up and all message related items that will be outputted by the shell. The shell in itself only calls kernel functions. So it will be constantly alluding kernel systemCall();

Here is an initial design:



In addition, what follows is a relationship graph between the classes contained in the prospective Shell:



Figure , Relationship outlook between classes

## 