NAME: MOHAN SANTOSH AMBEKAR

MIS NO: 111708007

SUBJECT: SPOS PROJECT

COLLEGE: COEP

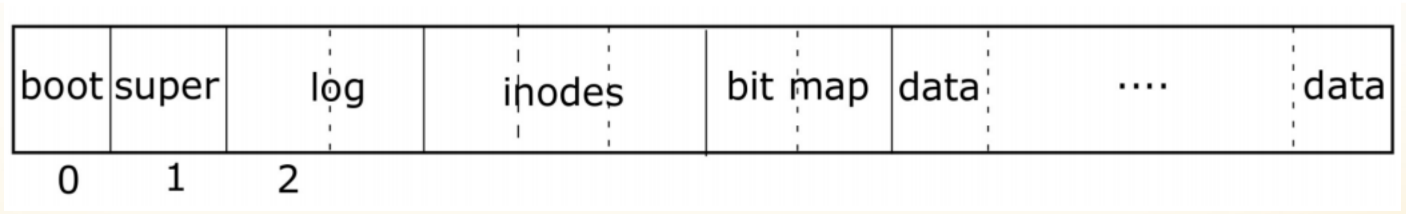
INTRODUCTION

Xv6 operating system

- Xv6 is a teaching operating system developed in the summer of 2006 for MIT's operating systems course.
- xv6 is a modern reimplementation of sixth edition unix in ANSI C for multiprocessor x86 and RISC-V systems.
- XV6 is a lightweight operating system.

File system in xv6

Layout of file system



- *Block 0, 1, 2 are fixed*
- Block 0: Boot code
- Block 1: Super Block, Store metadata about the file system
- Block 2: Log area, Use for transactions. Maintain consistency in case of a power outrage or system shutdown accidentally

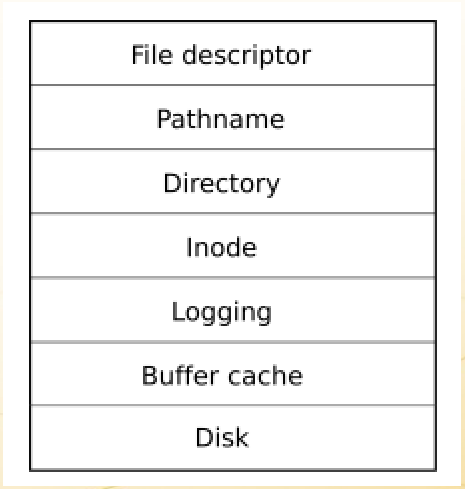
File system in xv6

- Inodes: Unnamed files
- Bitmap: An area to check which blocks are in use
- Data area: Actual data located

File system in xv6

Bottom up Approach

- File descriptors
- Recursive lookup
- Directory Inodes
- Inodes and Block allocator
- Logging
- Buffer Cache
- Disk



Topic

- The project is based on implementation of pwd command in xv6 operating system.
- pwd stands for print working directory.
- It prints the path of the working directory, starting from the root.

MOTIVATION

Need for pwd command

- Xv6 operating system consists of various system calls and functions.
- The commands like cd, mkdir, ls are already present in the system.
- But many times it is necessary to know the current working directory.
- A function like pwd is not present in the system and so there is a need of such a command.

PROJECT GOAL

Goal

- The goal of the project is to implement pwd command in the system without disturbing any other part.
- A system call is needed and pwd userspace program is also needed.
- The project deals with the file system topic.
- A few functions are needed to be added to implement this command in the shell.

Design Architecture

System call

- syscall pwd is assigned an id in syscall.h file.
- Then to syscall.c definition for system call is added.
- A pwd function is needed to be defined for user code as well.
- A header definition for the pwd function is added to user.h.
- To define the actual function, we need to use assembly (to issue the interrupt to switch to kernel mode). That's already done and defined as SYSCALL macro. A definition for pwd is added to usys.S file.

System call

- pwd, when called by user code, will use the definition in user.h file.
- This will be linked to the assembly function generated by the SYSCALL macro.
- This function moves the SYS_pwd constant into %eax and then issues the interrupt, switching to kernel mode.
- sysfile.c will include the implementation for system call.

System call

```
int  sys_pwd(void)  {
     char *p;
     int n;
     struct proc *curproc = myproc();
     if(argint(1, &n) < 0 || argptr(0, &p, n) < 0)
          return -1;
     return name_for_inode(p, n, curproc->cwd);
}
```

Inode

- In Xv6 (and most Unix file systems), the inode, short for 'index node' (though this may be a backronym), is a number pointing to a specific block on the disk that holds information about the file.
- On xv6, the inode struct stores only a few pieces of data. However, the inode is a layer of abstraction below the concept of the filesystem hierarchy and file names.
- The filesystem hierarchy is created with special inodes - type T_DIR - which contain a series of dirent structures. Each dirent is a tuple of a string name and an inode which references the file associated with that name.

Directory extraction

```
int name_of_inode(struct inode *ip, struct inode *parent, char
buf[DIRSIZ]) {
        uint off;
        struct dirent de;
        for (off = 0; off < parent->size; off += sizeof(de)) {
                if (readi(parent, (char*)&de, off, sizeof(de)) != sizeof(de))
                        panic("couldn't read dir entry");
                if (de.inum == ip->inum) {
                         safestrcpy(buf, de.name, DIRSIZ); return 0;
                }
        }
        return -1;
}
```

Directory extraction

- Directories in the xv6 filesystem are files whose contents are a just a series of dirent structures.
- Dirent is just a structure of a ushort inode id and a name.
- That means all this loop does is loop over every directory entry in the directory, loading it into de with readi.
- If the inode is not found -1 is returned. This happens when the file system is broken.

Directory extraction

```
int name_for_inode(char* buf, int n, struct inode *ip) {
    int path_offset;
    struct inode *parent;
    char node_name[DIRSIZ];
    if (ip->inum == namei("/")->inum) {
        buf[0] = '/';
        return 1;
    }
    else if (ip->type == T_DIR) {
```

Directory extraction

```
parent = dirlookup(ip, "..", 0);
ilock(parent);
if (name_of_inode(ip, parent, node_name)) {
 panic("could not find name of inode in parent!");
}
path_offset = name_for_inode(buf, n, parent);
safestrcpy(buf + path_offset, node_name, n - path_offset); path_offset
+= strlen(node_name);
if (path_offset == n - 1) {
      buf[path_offset] = '\0';
      return n;
 } else {
      buf[path_offset++] = '/';
 }
```

Directory extraction

```
iunlock(parent);
    return path_offset;
 }
else if (ip->type == T_DEV || ip->type == T_FILE) {
    panic("process cwd is a device node / file, not a
directory!");
} else {
    panic("unknown inode type");
}
}
```

Directory extraction

- namei is a wrapper which turns a full path into a inode.
- If the node is root buf is set to “\”.
- The inode types are defined in stat.h as T_DIR, T_FILE, and T_DEV - a device node.
- The parent reference is grabbed with dirlookup. ilock makes sure the inode is loaded from disk.
- Next, name_of_inode function is called and string manipulation is done.
- Finally the inode is released with iput and length of the path is returned

Testing code

```
#include "types.h"
#include "user.h"
#define MAX_PATH 512
 int main(int argc, char *argv[]) {
        char path[MAX_PATH];
        pwd(path, MAX_PATH);
        printf(0, "%s\n", path);
        exit();
}
```

OUTCOME

Result

- The pwd command was implemented properly with no errors.
- The command prints the complete path of the current working directory.
- The previous system gets added with one sytem call pwd .
- The function recursively derives the path of the current working directory.

Future work

- The pwd command works in all directories but the other commands like cd, ls, mkdir don’t work in the other directories except root directory.
- If these commands work in all directories then pwd will work at its best.