**2023**

**Operating System**

**4조**

|  |  |
| --- | --- |
| **Professor** | **김양우 교수님** |
| **Department** | **정보통신공학과** |
| **Member** | **2019112086 최재원**  **2019112110 이정민**  **2019112128 신예준**  **2020112084 제보민**  **2020112135 안세림** |
| **Due date** | **2023. 06. 06** |

**<목차>**

**1. 프로젝트 목표 및 방향**

(1) 추진목표

(2) 연구의 목적

(3) 연구과제의 필요성

(4) 관련연구

**2. 프로젝트 구현**

(1) 구조체와 main문

(2) 필수 명령어

ls, cat, chmod, grep, cd, mkdir, chown

(3) 추가 명령어

find, rm, cp

(4) 그 외의 함수

(5) 멀티 스레딩

**3. 자원 소요 계획**

(1) 프로젝트 팀원 역할 소개 및 참여기여도

(2) 프로젝트 추진 단계

(3) 전체 추진 일정

(4) 회의 일지

**4. 참고문헌**

**1. 프로젝트 목표 및 방향**

(1) 추진목표

c언어를 이용하여 리눅스 쉘 프로그램(ls, cd, mkdir, cat, chmod, chown, grep)을 설계한다.

(2) 연구의 목적

이번 프로젝트의 목적은 리눅스 시스템의 이해와 응용에 있다. 본 프로젝트는 C 언어를 통해서 리눅스 시스템을 구현함에 있다. 리눅스의 기본 동작 구조와 체계를 이해하고 동일하게 작동 가능한 시스템을 구현한다. 또한 AWS 환경을 사용하면서 클라우드 기본 동작 구조와 체계를 이해하고 동일하게 작동 가능한 시스템을 구현한다.

(3) 연구과제의 필요성

1. 이번 프로젝트의 목적은 웹 서버 구축과 활용을 통해 리눅스 서버의 사용의 이해도 향상과 리눅스 시스템의 이해와 응용에 있다.
2. 1학년때 기초 수업부터 배운 C언어라는 다소 익숙한 프로그래밍 언어를 이용해 리눅스에서 기본적인 작업 환경을 구축하고 리눅스와 리눅스 명령어를 직접 설계함으로써 리눅스라는 운영체제의 이해도를 높일 수 있다.
3. 프로세스와 스레드에 대해 이해할 수 있다.
4. window와 리눅스를 연결할 수 있는 secure shell 프로그램인 mobaxterm, putty에 익숙해질 수 있다.
5. 파일관련 함수들을 배움으로써 여러 가지 파일 관련 함수들을 익힐 수 있다.
6. 절대경로와 상대경로에 대해 이해할 수 있다.
7. 팀원끼리 모여 한 단위의 팀처럼 프로그래밍 프로젝트를 진행함으로써 미래의 개발자의 기본적인 자세와 소양을 갖출 수 있고 협력성과 리더십을 기를 수 있다.
8. AWS를 직접 사용하고 EC2 연결을 해보면서 학우 각각의 클라우드 서비스 활용 능력을 기르고 사용법을 익힐 수 있다.
9. 웹 서버와 리눅스를 연동함에 있어 리눅스 등 유닉스 계열이 아니더라도 윈도우나 노벨 넷웨어 운용을 배울 수 있다.

(4) 관련 연구

파일 디렉토리 형식을 구조체 형식으로 선언한 후 구조체에 대한 알고리즘을 연구하기 위한 관련 연구 논문과, 리눅스 명령어 관련 논문을 연구하였다.

(단행본: Linux unix shell apache programming. 2007. 컴스페이스 )

(해외 학술논문: Linux on the System . 2000. USENIX)

(해외 학술논문: The Linux BIOS. 2000. USENIX

**2. 프로젝트 구현**

(1) 구조체와 main문

|  |
| --- |
| 구조체 |
|  |

tageUserNode, tagUser, tagTreeNode, TagDirectoryTree, tagStackNode, tagStackd 구조체를 사용하였다. 이러한 구조체를 이용해 다른 함수들을 구현하였다.

|  |
| --- |
| main |
|  |

user들의 정보, 디렉토리, 스택을 이용해 리눅스 환경에서 login하고 명령어를 실행시킨다.

(2) 필수 명령어

|  |
| --- |
| ls |
|  |

ls는 디렉토리에 있는 모든 내용(디렉토리, 파일 등)을 확인하고 제공하는 명령어이다. Ptr->Ichild를 하였을 때 NULL 값인 경우 디렉토리가 비었다 라는 출력문이 뜨도록 구현하였다. 그 외에 NULL 값이 아닐 경우, 디렉토리의 모든 내용이 뜨도록 구현하였다. If(strlen(prt->name)<8)은 내용에 대한 줄을 출력했을 때, 이 줄이 더욱 보기 좋고 깔끔하게 출력될 수 있도록 구현하였다. ls에서는 ls함수, ls-l함수, ls-a함수, ls-al함수로 나누어 구현하였다.

|  |
| --- |
| ls |
|  |

터미널에 ls 명령어를 입력해 파일이나 디렉토리등 모든 내용을 표시하였다.

|  |
| --- |
| ls-a |
|  |

ls-a 함수는 숨겨진 파일이나 디렉토리를 출력해준다. Lchild가 없었다면, directory empty를 출력하고, RightSibling이 NULL값이 아닐 경우엔 tmpNode의 숨겨진 파일이나 모든 디렉토리를 출력한다. 출력할 때 역시 If(strlen(prt->name)<10)을 통해 깔끔하게 디렉토리가 출력되도록 구현하였다.

|  |
| --- |
| ls-a |
|  |

ls-a 명령어를 사용하여 숨겨진것을 포함한 모든 파일과 디렉토리를 출력하였다.

|  |
| --- |
| ls-l |
|  |

ls-l은 ls에 대한 자세한 내용을 출력한다. Lchild가 NULL값일 경우 direct empty를 출력한다. Lchild가 NULL값이 아닐 경우 Lchild의 rsibling이 NULL값이 아니면 rsibling으로 이동하게 된다. 그 외에 type, size 등 해당되는 모든 정보를 출력한다. Rsibling이 NULL값일 경우 rsibling으로 이동하지 않고 현재 node에서 모든 자세한 내용을 출력한다.

|  |
| --- |
| ls-l |
|  |

ls-l 명령어를 사용하여 퍼미션(권한), 포함된 파일수, 소유자, 그룹, 파일크기, 수정일자, 파일이름을 표시하였다.

|  |
| --- |
| ls-al |
|  |
|  |

ls-al은 모든 디렉토리의 내용을 출력한다. Lchild가 NULL값일 경우 directory empty를 출력한다. Lchild가 NULL값이 아닐 경우 rsibling과 lchild의 내용을 모두 출력한다. (.)의 경우 본인 디렉토리의 정보를 read해 출력하고, (..)의 경우 상위 디렉토리의 정보를 read해 출력한다.

|  |
| --- |
| ls-al |
| get\_UID, get\_GID |
|  |

get\_UID, get\_GID 함수를 통해서 user ID와 group ID를 얻는다. 특이점으로 가독성을 향상 시키기 위해서 string으로 출력되게 했다..

|  |
| --- |
| ls-al |
|  |

|  |
| --- |
| cat |
|  |
|  |
|  |
|  |
|  |
|  |

Cat 명령어는 파일 또는 텍스트 파일의 작은 내용을 생성하거나 화면(터미널)에 출력하는 명령어이다. 처음으로 Haspermission 함수로 쓰기 권한이 있는지 없는지 확인을 한다. 쓰기 권한이 없다면 error message로 ‘권한 없음’을 띄우고 쓰기 권한이 있다면 IsExistDir(dirTree, str, ‘d’)함수를 통해 type=d면 디렉토리인지 파일인지 확인한다. IsExistDir의 값이 NULL인 경우에는 찾으려는 이름의 파일이 있거나 없는 경우로서 Concatenate(dirTree, str, 0)함수를 실행시킨다.

Concatenate함수는 fopen을 통해 쓰기 권한으로 파일을 불러오고 난 후 키보드로 입력 받은 값을 입력해준다. 입력을 종료하려면 리눅스에서 Ctrl+D를 입력하면 된다. Cat “>” 옵션은 파일이 있다면 기존 파일에 내용을 덧붙이고, 파일이 없다면 새로 만들어 주는 옵션이다. IsExistDir(dirTree, fNam, ‘f’)함수를 실행시켜 같은 파일의 존재 유무를 확인시켜준다. 존재한다면, 시간 값들을 최신화 시키고, 파일이 존재하지 않는다면, MakeDir(dirTree, fName, ‘f’)를 통해 새로운 파일을 만들어주다. “-n” 옵션은 파일을 줄 별로 번호를 메겨 출력하고, 옵션이 없고 텍스트 파일의 이름만 있다면 텍스트 파일 출력하기를 실행한다.

|  |
| --- |
| Cat |
|  |

cat 명령어를 이용하여 텍스트 파일을 추가하였다. “>”옵션으로 새로 파일을 만들었고 내용을 입력한 후 Ctrl+D를 입력하여 저장해주었다. “-n” 옵션을 이용하여 file1.txt를 줄 별로 번호를 매겨 출력하게 하였다. 옵션을 지정하지 않은 file.txt는 텍스트 파일을 출력하였다.

|  |
| --- |
| Cat thread |
|  |
|  |

Cat Thread함수는 받은 쓰레드의 데이터와 경로 그리고 cmd, option을 저장한다. 그 외 조건을 판정하기 위해서 변수들을 선언한 후, 옵션이 있을 경우와 없을 경우 두가지로 나누어서 진행한다. 이때 파일의 권한을 비교하여 권한이 있을 경우 경로 데이터를 통해 파일의 경로와 타입을 판정하고, 이때 문제가 없다고 판단되면, concatenate함수로 데이터를 전송하여 cat함수를 실행한다. 또한 옵션이 없을 때에도 마찬가지로 작동하며 concatenate로 보내주는 옵션의 데이터만 다르게 전달한다. 해당 함수의 기능이 끝나면 쓰레드를 종료한다.

|  |
| --- |
| chmod |
|  |
|  |
|  |

chmod 명령어는 파일의 mode를 변경하는 명령이다. 즉, 파일 및 디렉토리의 접근 권한을 변경하고 싶을 때 사용한다. R옵션을 통해 하위 디렉토리 및 파일들의 접근 권한까지 모두 변경한다. 위에서는 예외처리 하고, tmpNode를 통해 directory의 존재를 확인한다. 값이 NULL이 아닌지 확인하고 만약 NULL 값을 가지지 않는 것이 있다면 해당 권한 값을 변경한다 해당 디렉토리의 하위 파일 혹은 접근 권한 까지 다 바꾸는 옵션이 NULL일때까지 재귀를 사용한다.

|  |
| --- |
| Chmod |
|  |

chmod 명령을 사용하여 해당 폴더나 파일의 권한 속성을 변경 할수가 있다.

모드의 종류는 크게 세 가지인 'R, W, X'로 나뉜다.

R = Read (읽기 가능)

W = Write (쓰기 가능)

X = Excute (실행 가능)

777의 권한이 부여되었다면 각각의 7은 (4 + 2 + 1), 즉 읽고 쓰고 실행 모두 가능하며 각각의 7은 첫자리부터

처음의 7 - 소유자

두번째의 7 - 그룹 사용자

세번째의 7 - 기타 사용자

위와 같이 나타난다. 위의 조건에 따라 “권한을 부여한 뒤 ls-l 명령어를 통해 “test3”가 “drwxrwxrwx”로 변한 것을 확인할 수 있었다.

|  |
| --- |
| Chmod thread |
|  |

Chmod Thread는 threadTree를 통해서 데이터를 받은 후, 경로는 dirTree를 통해 저장한다. mod와 cmd의 데이터를 저장한 후, ChangMode 함수로 전송하여, mod를 변경한 후 쓰레드를 종료한다.

|  |
| --- |
| grep |
|  |
|  |
|  |
|  |
|  |
| grepprint |
|  |

Grep명령어는 입력으로 전달된 파일의 내용에서 찾고자 하는 패턴을 입력으로 받아 특정 패턴에 대해 지정된 파일을 검색하는 명령어이다. Grep을 구현하는데 잘못된 옵션 입력에 관련하게 더욱 신경을 썼다. 예를 들어 단일 옵션 2개가 동시에 들어올 경우를 생각해 단일옵션 -i, -v가 동시에 들어왔을 때 코드를 구현하였다.

|  |
| --- |
| grep |
|  |

file에 어떤 내용이 있는지 출력해보았다. 그 후에 grep 명령어를 사용해 file내에 “file”이라는 문자가 있는 문자열을 출력하였다. “-n” 옵션을 사용해 file내에 “file”이라는 문자가 있는 문자열의 줄에 번호를 매겨 출력하였다. “-v” 옵션을 사용해 file내에 “file”이라는 문자가 없는 문자열을 출력하였다. “-i” 옵션을 사용하면 대소문자 구분을 하지 않고 그 문자가 있는 문자열을출력해주기 때문에 문자를 대문자로 설정하여 “FILE”로 검색을 하였음에도 소문자인 “file”도 같이 검색되어 나온 것을 알 수있다. “-i”와 “-v”옵션이 합쳐진 “-iv” 옵션을 이용하여 대소문자 구분없이 “FILE”의 문자가 없는 문자열을 출력하였다. “-i”와 “-n”, “-v”옵션이 합쳐진 “-inv” 옵션을 사용하여 대소문자 구분없이 “FILE”의 문자가 없는 문자열의 줄에 번호를 매겨 출력하였다.

|  |
| --- |
| grep thread |
|  |
|  |

Grep Thread함수는 받은 데이터를 이용해서 directory와 cmd 등의 데이터를 받고 난 후, User의 권한을 확인한다. 그리고 이상이 없을 시 해당 파일에 관한 조건문을 설정한다. 그리고 파일 위치와 속성과 관련해서 문제가 없을 경우, grepPrint함수로 데이터를 전송 하여 grep함수 기능을 실행한다. 이후 작업을 마친 쓰레드는 종료한다.

|  |
| --- |
| cd |
| MoveCurrent |
|  |

|  |
| --- |
| cd |
| MovePath |
|  |

현재 디렉터리에서 명령자가 원하는 디렉터리로 옮기는 함수. 디렉터리가 root일경우 root로, 절대적인 디렉터리일 경우 원하는 디렉터리로 옮겨준다. 상대적인 디렉터리일 경우엔 원하는 디렉터리로 옮겨주나, 디렉토리가 없으면 현재 노드로 되돌아간다.

|  |
| --- |
| Cd |
|  |

먼저 pwd 명령어를 사용하여 현재 위치를 파악한 후 cd 명령어를 사용하여 현재 작업 디렉토리에서 “home”이란 작업 디렉토리로 작업 경로를 이동하였다. 그리고 cd .. 명령어를 이용하여 현재 디렉토리에서 상위 디렉토리로 이동하였다.

|  |
| --- |
| mkdir |
| MakeDir |
|  |
|  |

Makedir함수는 노드에 time, size, mode, type등을 추가하게 하였다. 이때 자신 노드에서 type이 자기자신일 때 mode = 700, SIZE = 4096, d일 때 mode = 755, SIZE = 4096, f일 때 mode = 644, SIZE = 0을 추가한다. 이후에 Mode2Peission을 사용해서 newnode에 UID, GID, month, day, hour 등과 leftchild와 rightsibling노드를 추가한다.

|  |
| --- |
| mkdir |
|  |
|  |

Mkdir 명령어의 p(parents) 옵션은 상위 폴더가 없으면 만들어주는 기능을 수행한다.

Mkdir함수는 새로운 디렉터리를 생성하는 명령어이다. 앞서 설명했던 MakeDir함수를 사용한다. -을 입력할 경우, 만약 경로가 없으면 ‘Invalid option’, 경로가 root노드이면 /를 하나만 출력하도록, 경로가 null이 아니면 movecurrent함수와 makedir(type d)를 이용해서 새로운 디렉터리를 생성한다. -m을 입력할 경우, 경로가 없으면 잘못된 연산자라는 출력이, 그 외에는 makeDir를 실행한다.

|  |
| --- |
| Mkdir |
|  |

ls -l 명령어를 사용하여 현재 파일과 디렉토리들을 출력한 후 mkdir명령어를 사용하여 “new\_dir”이라는 디렉토리를 추가해준 뒤 ls-l 명령어를 통해 추가되었는지 확인해보았다. 또 “-p” 옵션을 사용하여 “test3” 디렉토리를 추가할 때 상위 경로도 함께 생성해주었다.

|  |
| --- |
| mkdir thread |
|  |
|  |

mkdir Thread함수는 받은 데이터를 통해 dirTree에 저장한다. 그러고 난 후, 사용할 변수들을 선언해주고 if문을 통해 경로를 판정한다. 상대경로와 절대경로로 나누어서 파일을 생성하고, 옵션선택이 되어있을 경우, 경로를 진행하는 중, 없는 directory는 생성한다. 그리고 옵션 선택이 안되어 있을 경우, 중간 경로의 directory가 없을 시 종료한다. 해당 작업이 끝난 이후에는 쓰레드 종료를 선언한다.

|  |
| --- |
| chown |
|  |
|  |

Chown은 파일이나 디렉토리의 소유자 및 권한을 변경하는 명령어이다. R옵션을 통해 하위 디렉토리 와 파일들의 접근권한까지 모두 변경한다. User가 있으면 tmp에 정보를 담는다. 디렉토리를 바꾸려 그 디렉토리가 있으면 changeowner와 changeownerall함수를 구현한다. 허가가 나지 않는다면 파일을 수정할 수 없다는 결과창을 띄어준다. User name이 있는지도 확인한다. 오류가 난다면 접근할 수 없다는 결과창을 띄어준다.

|  |
| --- |
| Chown |
|  |

ls-l 명령어를 사용해 각 디렉토리와 파일들의 소유자를 확인한 후 chown 명령어를 사용하여 “test3”의 소유자를 “root”에서 “os”로 변경한 후 다시 ls-l 명령어를 사용하여 소유자가 제대로 변경이 되었는지 확인했다.

|  |
| --- |
| chown |
| ChangeOwner |
|  |

changeownerall함수는 UID와 GID를 복사한다.

|  |
| --- |
| chown thread |
|  |

Chown thread 에서 ThreadTree 함수로 쓰레드 데이터를 받는다, Cmd와 ID 정보를 저장하고 난 후, 입력된 문자열에 오류가 없으면 ChangeOwner함수를 실행한다. 그리고 해당 기능이 완료된 후 해당 스레드를 종료한다.

(3) 추가 명령어

|  |
| --- |
| find |
|  |

Find는 주어진 조건에 따라 디렉터리를 검색해서 원하는 파일을 찾는 명령어이다.

|  |
| --- |
| Find |
|  |

find에 “-name”을 이용하여 “dd”이라는 파일을 검색하였다.

|  |
| --- |
| find |
| ReadDir |
|  |

ReadDir 함수는 FindDir로 열었던 directory.txt파일을 띄어쓰기 단위로 str변수에 입력한다. 그후 strstr함수를 통해 경로를 출력하며 찾고자 하는 파일/디렉토리를 찾는다.

|  |
| --- |
| find |
| FindDir |
|  |

FindDir함수는 읽기 전용으로 디렉토리 텍스트 파일을 열어서 줄 바꿈 단위로 ReadDirg함수를 불러오는 하는 함수이다

|  |
| --- |
| rm |
| RemoveDir |
|  |

tempNode가 LeftChild에 위치해 있는 경우 LeftChild가 NULL값인 경우 “no such file or directory”라는 문구가 출력되도록 설정하였고 NULL값이 아닐땐 tempNode를 RightSibling을 사용하여 LeftChild로 지정한 후 tempNode를 DelNode로 설정하고 현재 노드 포함해서 하위 노드들까지 다 삭제한다. 만약 DelNode가 NULL이면 중간에 파일이나 디렉터리가 지워질 수도 있으므로 끊어지지 않도록 prevNode->RightSibling = DelNode->RightSibling를 통해 이어주었다.

|  |
| --- |
| rm |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

rm은 빈 디렉터리를 삭제할 때 사용하는 명령어. 옵션이 r일 때, f일 때, rf일 때로 나누어서 구성했다. r일때에 명령이 NULL일 경우 ‘Try 'rm --help' for more information.’라는 문구가 출력되고, -r일떄는 하위 디렉터리 삭제, -f일때는 강제 삭제를, -rf는 옵션 r과 f의 기능을 합친 것으로 디렉터리도 삭제하고 강제 삭제도 구현하였다. -r일때에는 IsExistDir 함수를 이용해서 디렉터리가 존재하는지 확인하고 존재하면 앞서 설명했던 RemoveDir함수를 이용해서 디렉터리를 지운다. -f일때에는 IsExistDir함수를 사용해서 파일과 디렉터리의 존재를 확인하고 tempNode와 tempNode2에 할당한 후 RemoveDir함수를 통해 강제삭제를 하도록 하였다. -rf는 위에서 설명했듯이 디렉터리와 강제삭제가 모두 가능하도록 코드를 구성하였다. 이외에 –help일때에는 rm의 옵션에 대해 설명해주는 설명문이 출력되도록 구성하였다.

|  |
| --- |
| rm |
|  |

ls명령어를 사용하여 모든 디렉토리와 파일들을 출력한 후 rm 명령어를 사용하여 “file.txt”를 삭제한 후 ls 명령어를 사용하여 제거가 되었는지 확인했다. 옵션 “-r”를 사용하여 디렉토리를 삭제했고, 옵션 “-f”를 사용하여 강제 삭제를 하였다. 옵션 “-r”과 “-f”를 합친 옵션 “-rf”를 사용하여 디렉토리를 강제 삭제 한 뒤 ls명령어를 통해 모든 명령어가 제대로 수행 되었는지 확인해주었다.

|  |
| --- |
| cp |
|  |

Cp는 copy의 줄임말로 무언가를 복사할 때 사용하는 명령어이다. 즉, 복사 작업을 진행할 디렉토리와 결과물 파일명, 원본파일명을 인수로 받아 입력 받은 두 파일명을 우선 출력하는 것이다. Access와 strcmp를 이용하여 입력 받은 파일명의 유효성을 검사한다. open함수의 oflags를 사용하여 파일이 없으면 새로 생성한다. 그러고 나서 마지막으로 write를 호출하여 buf의 내용을 field 파일에 기록하고, nread에 기록된 byte를 기록한다. 그 후 Makedir를 호출하여 f형식으로 기록한다.

|  |
| --- |
| Cp |
|  |

cat 명령어를 통해 텍스트 파일 “file1.txt”를 추가해준 후 cp 명령어를 사용하여 “file1.txt”를 “file7.txt”라는 이름으로 현 위치에 복사하였다. ls명령어를 통해 “file7.txt”가 제대로 복사되었는지 확인하고 cat명령어를 통해 내용도 복사 되었는지 확인하였다.

(4) 그 외의 함수

|  |
| --- |
| pwd |
|  |
|  |

Pwd는 현재 어떤 디렉토리 경로에 있는지 위치를 절대경로로 표시하는 알려주는 명령어이다. Printpath 함수로 stack의 경로가 표시되도록 구현하였다. NULL값을 입력하였을 경우, printpath 함수를 이용하여 디렉토리의 위치를 알려주도록 구현하였고, --help 시, 도움말이 출력되도록, - 시, 잘못된 연산자와 부적절한 옵션이 출력되도록 구현하였다.

|  |
| --- |
| user |
| WriteUser |
|  |

Writeuser함수는 로그인한 아이디 시간을 최신화 하고 난 후 현재 user의 정보를 저장한다. 재귀를 통해 linknode의 tail까지 read하는 알고리즘을 사용한다.

|  |
| --- |
| user |
| SaveUserList |
|  |

Saveuserlist함수는 로그인 정보를 반영해주는 함수이다.

|  |
| --- |
| user |
| ReadUser |
|  |
|  |

Readuser함수는 user.txt로 정보가 들어오게 되면 각각의 정보를 NewNode에 mapping해주는 역할을 수행하는 함수이다. root라면 tail와 head을 linkedlist로 만들어 잇고 그 외에는 tail의 linknode를 linkedlist로 만들어 잇는다.

|  |
| --- |
| user |
| LoadUserList |
|  |

Loaduserlist함수는 User.txt 일기 권한으로 열고 while 문의 fgets함수로 tmp에 저장하고 그 유저에 대한 정보로 readuser함수를 실행한다.

|  |
| --- |
| user |
| IsExistUser |
|  |

IsExistUser함수는 존재하는지 user 인지 확인해주는 함수이다. returnUser=userList->head로 하고 NULL값일 때까지 while문을 실행시킨다. 같은 uerName이 존재한다면 break후 returnUser를 반환시킨다.

|  |
| --- |
| Permission |
| HasPermission |
|  |
|  |

Haspermission함수는 소유자권한(UID), 그룹사용자권환(GID), 기타 사용자 권한을 확인해 주는 함수이다. 소유자권한: dirNode의 permission[8]배열의 0, 1, 2번째 index가 각 0이라면, 권한이 없고 1이라면, 있다는 정보를 갖는다. 그룹사용자권한: dirNode의 permission[8]배열의 3, 4, 5번째 index가 정보를 갖는다. 기타사용자권환: dirNode의 permission[8]배열의 3, 4, 5번째 index가 정보를 갖는다.

|  |
| --- |
| Permission |
| Mode2Permission |
|  |

Mode2permission은 입력 받은 권한에 대한 정보 각각의 숫자를 /와%연산을 통해 3bit 이진수로 변환해, permission을 바꿔주는 역할을 수행하는 함수이다. 이후 각각 권한을 r w x 로 출력 하는 것은 printpermission함수가 수행할 것이다.

|  |
| --- |
| Permission |
| PrintPermission |
|  |

Printpermission함수는 각각의 권한을 r w x 로 출력 할 수 있게 변환해주는 함수이다.

|  |
| --- |
| Stack |
| IsEmpty |
|  |

IsEmpty함수는 stack이 비어있는지 확인해주는 함수이다.

|  |
| --- |
| Stack |
| InitializeStack |
|  |

InitializeStack함수는 stack을 생성하는 함수이다.

|  |
| --- |
| Stack |
| Push |
|  |

sPush함수는 stack에 쌓은 후 tmpNode의 Parent로 계속 이동한다.

|  |
| --- |
| Stack |
| Pop |
|  |

Pop함수는 stack에 쌓인 정보를 순서대로 empty가 될 때까지 경로를 출력하게 해준다.

|  |
| --- |
| Time |
| GetMonth |
|  |

|  |
| --- |
| Time |
| GetWeekDay |
|  |

|  |
| --- |
| Directory |
| IsExistDir |
|  |

IsExistDir함수는 존재하는 디렉토리 또는 파일인지 확인해 주는 함수이다.

|  |
| --- |
| Directory |
| GetDir |
|  |

getDir함수는 상대경로일 때도, 각 명령어가 기능을 수행할 수 있게 하기 위한 함수이다. Movepath함수를 적용해 경로를 이동한 후 상대경로를 수행하는 명령어들을 수행할 수 있도록 함으로써 command가 수행될 수 있게 구현하였다.

|  |
| --- |
| Directory |
| GetPath |
|  |

getPath함수는 tmpNode를 dirNode의 Parent node로 설정하고 tmpNode가 root이라면, /를 출력하고 아니면 디렉토리가 root일 때까지 push를 반복하게 하여 stack이 empty 될 때까지 pop한다

.

|  |
| --- |
| Directory |
| WriteNode |
|  |

WriteNode함수는 SaveDir함수를 통하여 Directory.txt를 쓰기 권한으로 열고 정보를 text에 담고 save하는 역할을 수행하는 함수이다.

|  |
| --- |
| Directory |
| ReadNode |
|  |
|  |

ReadNode함수는 LoadDir함수에서 fgets로 한줄 씩 입력 받은 후 입력 받은 정보를 각 매칭되는 node에 mapping하는 역할을 해주는 함수이

|  |
| --- |
| Instruction |
|  |
|  |
|  |

Instruction함수는 입력 받은 strcmp와 command 함수를 통하여 구현했던 멍령어와 비교해, 어떤 명령어를 실행시킬지 정해주는 함수이다

|  |
| --- |
| GetInformation |
| Login |
|  |

Login 함수는 리눅스에 접속할 때 ID를 입력하여 접속하는 것과 유사하게 가능하도록 만든 함수이다. fget함수로 키보드에서 입력하여 배열에 UserName을 입력하고 그 입력 받은 UserName이 리스트에 있는지 확인한다.

|  |
| --- |
| GetInformation |
| Printhead |
|  |
|  |

PrintHead함수는 우분투 터미널처럼 명령어 입력 전에 출력되는 같은 양식을 계속해서 반복적으로 출력해주는 함수이다.

|  |
| --- |
| Save |
| IntializeTree |
|  |

Tree를 생성해주는 함수이다. Newnode를 생성하고 Newnode에 type,mode,UID,GID,SIZE,month, day,hour,minute 등 추가할 있는 node를 만들고 tree를 생성한다.

|  |
| --- |
| Save |
| SaveDir |
|  |

SaveDir 함수는 Directory..txt를 쓰기 권한으로 열고 WriteNode함수를 통해 정보를 입력한다.

|  |
| --- |
| LoadDir |
|  |

LoadDir함수는 load기능을 수행하는 함수이다.

Directory.txt를 읽기권한으로 열어 fgets함수를 통해 한줄씩 ReadNode 함수를 실행시킨다.

(5) 멀티 스레딩

|  |
| --- |
| multithreading |
|  |

Cpu는 여러 개의 스레드를 처리할 수 있는 자원을 가지고 있다. 멀티 스레드는 이러한 cpu의 최대 활용을 위해 프로그램의 두개 이상을 동시에 실행하게 해주는 기술이다. 멀티 스레드의 장점에는 첫번째로, 프로그램의 스레드 중 하나가 긴 작업을 하거나 중단이 되더라도, 프로그램이 계속 수행 되어 사용자에 대한 응답성이 증가한다. 두번째로, 프로세스 내의 여러 메모리 및 자원들을 공유하기 때문에 시스템 자원의 소모가 줄어들고 메모리 공간도 줄어든다. 그로 인한 경제성이 좋다고 볼 수 있다. 마지막으로 다중 cpu 구조에서는 각 스레드가 다른 프로세서에서 병렬로 수행될 수 있어 병렬성이 증가해 멀티프로세서의 활용도가 높다. 그러나 임계 영역에 있어 공유하는 자원에 동시에 접속하는 경우, 프로세스와는 달리 데이터와 힙 영역을 공유하기 때문에 A스레드가 B스레드에서 사용 중인 자료구조나 변수에 접근하여 원하는 값이 아닌 전혀 다른 엉뚱한 값을 읽어 오거나 수정할 수 있는 단점이 있다. 그리고 컨텍스트 스위칭과 동기화 등의 이유로 멀티 스레딩은 스레드 생성 시간이 오히려 오버헤드로 작용할 수 있어 단일 스레드보다 느리다.

Mkidr 명령어에서 다수의 폴더 생성 멀티스레딩을 이용하여 동시에 생성해야 하기 때문에 멀티스레딩을 구현하였다. 이 외의 명령어(cat, chmod, chown, grep)에서도 동시 작업이 발생해 멀티스레딩을 적용하였다. 이 작업을 할 때 어느 작업에서 동시작업이 발생하는지에 대하여 중심으로 연구하여 구현하였다.

|  |
| --- |
| Header, Thread |
|  |
|  |
| Pthread\_create() Pthread\_join()- ex)mkdirThread |
|  |

Cat, chmod, chown, grep 함수 내에서 멀티스레딩을 구현하였다. 리눅스에서 c언어로 pthread.h 헤더를 통해 pthread\_t를 이용하여 스레드의 개수를 선언 받았다. Pthread\_create를 이용해서 쓰레드를 생성한다. 스레드는 메인함수가 종료되면 자동으로 종료된다. 따라서 메인에서 스레드가 종료될 때까지 메인의 종료를 기다려야 하기 때문에 Pthread\_join을 사용했다. 위에 첨부 사진은 mkdirthread의 예제이지만 총 구현한 5개도 마찬가지로 위와 같다. Grep은 한 문장을 두개 이상의 디렉토리나 파일에서 찾기 위해 멀티스레드를 구현해야 한다. Grep은 유저의 권한을 확인하기 위해 디렉토리와 cmd를 받아야 해서 멀티스레드를 사용했다. 파일이 존재하는 것을 확인하고 해당 파일별로 쓰레드에 작업을 할당 시키기 위해 grep 멀티스레드를 사용했다. Mkdir은 두 개 이상의 디렉토리나 파일을 생성하기 위해 멀티스레드를 구현하였다. 경로를 판정하기 위해 상대경로와 절대경로로 나누어서 파일을 생성한다. 옵션이 선택 되어있을 경우 경로 진행중 없는 디렉토리를 생성하고, 옵션 선택이 안 되어있을 경우와 중간경로 디렉토리가 없으면 종료한다. 여기서 상대경로와 절대경로 파일생성 및 옵션 선택에 있어 디렉토리를 생성해야 함에 멀티스레딩을 사용했다. Mkdir -p asd/qwe: asd 안에 qwe 디렉토리를 생성한다. Cat은 IsExistDir함수를 실행시켜 같은 파일의 존재하는지 유, 무에서 존재하지 않는다면 Makedir을 통해 새로운 파일을 만들어주기 위해 멀티스레딩을 구현했다. Chmod는 파일의 mode를 변경하는 명령어 이기 때문에, 멀티스레드를 사용해서 하위 폴더를 하나 하나씩 mode를 싹 다 바꿔주었다. Chown은 파일이나 디렉토리의 소유자 및 권한을 변경하는 명령어인데, 멀티스레드로 하위 폴더를 하나 하나씩 소유자를 싹 다 바꿔주었다.

**3. 자원 소요 계획**

(1) 프로젝트 팀원 역할 소개 및 참여기여도

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **학번** | **2019112086** | **2019112128** | **2019112110** | **2020112135** | **2020112084** |
| **이름** | 최재원 | 신예준 | 이정민 | 안세림 | 제보민 |
| **업무분담** | 팀장으로써 인프라 구성 및 모든 업무 총괄 및 관여 | 보고서 작성 | 코드 작성 | 보고서 작성 | PPT 작성  및 발표 |
| **참여 기여도** | 20% | 20% | 20% | 20% | 20% |

(2) 프로젝트 추진 단계

1. 프로젝트 파악, 계획 수렴
2. 필요한 명령어 역할 분담 및 정보 수집
3. 명령어 설계 및 일괄 취합, 추가적인 옵션 구현 및 리뷰
4. 오류 수정 및 동작 확인
5. 보고서 작성 및 발표 ppt 자료 작성
6. 제출 및 발표

(3) 전체 추진 일정

|  |  |
| --- | --- |
| 5. 11 (목) | 프로젝트 계획 수렴 및 역할 분담, 다음 주 회의 전까지 숙제 정하기  (명령어 구현) |
| 5. 18 (목) | 저번 주 숙제 구현 코드 리뷰, 다음 주 회의 전까지 숙제 정하기  (코드 수정 및 추가 명령어 구현, 보고서 초안, ppt 초안) |
| 5. 25 (목) | 저번 주 숙제 리뷰(보고서, ppt 초안 및 코드 분리), 다음 주 회의 전까지 숙제 정하기 (EC2 사용법 숙지, 보고서, ppt 완성, 발표자료 작성) |
| 6. 1 (목) | 저번 주 숙제 리뷰(EC2 사용법, 보고서 ppt 확인, 발표자료 작성), 다음 주 마지막 숙제 정하기(명령어 사용법 완벽히 숙지, ppt, 보고서 완성) |

(4) 회의일지

|  |  |
| --- | --- |
| 5. 11 (목) | \*팀 프로젝트 진행을 위한 회의 일정 조율 – 매주 목요일 OS 실습 후  \*금일 회의 주제: 역할 분담(코드 작성, 발표, 보고서 작성, 인프라 구성)  코드작성(이정민) 발표(제보민) 보고서(신예준, 안세림) 인프라구성(최재원)  \*다음 주 회의 숙제- 코드 작성 인원들 맡은 명령어 구현  \*다음 주 회의 주제- 코드 작성 리뷰 및 추가 명령어 회의 |
| 5. 18 (목) | \*구현 코드 리뷰- grep, chown, cat, ls, chmod 등  \*금일 회의 주제- 추가적으로 구현할 내용  1. pwd, find, cp  2. cat control d 해서 파일 생성 멈춤  3. grep i, v, n  4. mkdir -m, -p  5. rm -rf로 파일도 삭제/현재 디렉토리만 되는 상태  \*이번 주 내로 해올 것-  1. C코드 header, C파일 나누기 -> link tree로  2. 보안그룹 설정 EC2  \*이번 주 내로 하면 좋은 것- mkdir rm -rf 시 파일도 삭제  \*다음 주 회의 숙제- 보고서 초안, ppt 초안 |
| 5. 25 (목) | \*저번 주 숙제 리뷰- 보고서, ppt 초안 및 코드 분리  \*금일 회의 주제-  1. 보고서 추가 내용(코드 구현 방법 작성, 및 검토)  2. EC2 보안그룹 설정  3. ppt 초안 검토 및 개선사항 수정 및 확인  \*이번 주 내로 하면 좋은 것- EC2 내에 git clone으로 파일 옮긴 후 makefile 설치  \*다음 주 회의 숙제  1. EC2 사용법 숙지  2. 보고서 , ppt 완성  3. 발표자료 작성 |
| 6. 1 (목) | \*저번 주 숙제 리뷰- EC2 사용법 확인, 보고서와 ppt를 확인하고 수정사항 개선, 발표자료 작성  \*금일 회의 주제-  1. 보고서 수정(thread 관련 설명 추가)  2. 발표자료 수정(명령어 설명 개선사항 수정)  3. 시연 대비 명령어 숙지(코드 시연 관련 명령어 실행 실습하며 숙지)  \*다음 주 마지막 숙제-  1. 명령어 사용법 완벽히 숙지  2. ppt, 보고서 실행 화면 및 명령어 설명 수정해서 완성 |

**4. 참고문헌**

1. https://www.youtube.com/playlist?list=PLfth0bK2MgIa6w63IglYQD\_qljDntSh-H

AWS강좌 EC2 기초, Amazon Aurora(아마존 오로라), Amazon Aurora 심화

AWS 보안 그룹 정리, Security Group, EBS/AMI로 EC2 백업

1. https://www.youtube.com/playlist?list=PLfth0bK2MgIan-SzGpHIbfnCnjj583K2m

클라우드 컴퓨팅종류, AWS 계정 만들기, EC2 유형과 크기, VPC와 Subnet, Amazon S3

1. 전문가를 위한 C 저자 캄란 아미니
2. OS 수업 PPT 자료
3. LCRS 트리 구조

https://en.wikipedia.org/wiki/Left-child\_right-sibling\_binary\_tree

https://www.geeksforgeeks.org/left-child-right-sibling-representation-tree

----------------------------------------------------------------------------------------------------

AWS 계정 ID/PW

**팀장 ID/ PW**

최재원 cjaewon0217@naver.com Dd9510010!

**팀원 ID/ PW**

이정민 [ljmtt2000@gmail.com](mailto:ljmtt2000@gmail.com) Dlwjdals6wjd!

신예준 shinminseouk@naver.com Tlsalstjr1@

제보민 [jbm0515@gmail.com](mailto:jbm0515@gmail.com) Top911349@

안세림 [ahnselim@naver.com](mailto:ahnselim@naver.com) selim0924!