
운영체제 과제2 보고서



과목: 운영체제 3분반
학과: 모바일시스템공학과
학번: 32224020
이름: 전은지

Index

1. Introduction

2. Background information

2.1 파일 시스템의 기본 개념

2.2 파일 시스템의 중요성

2.3 파일 시스템 시뮬레이터의 종류와 특징

3. Program description

4. Result

5. Conclusion

1. Introduction

이번 프로젝트의 목표는 간단한 파일 시스템 시뮬레이터를 구현하여, 파일 생성, 삭제, 읽기, 쓰기 뿐만 아니라 디렉토리 생성, 삭제, 이동, 그리고 파일 검색 기능까지 구현하는 것이다. 사용자는 다양한 명령어를 통해 파일과 디렉토리를 관리할 수 있다. 학생은 파일 시스템의 동작 원리를 이해하고, 이를 통해 파일 시스템의 기본적인 작동 방식을 학습하는 것이다.

2. Background information

2.1 파일 시스템의 기본 개념

파일 시스템은 컴퓨터의 저장 장치에서 데이터를 관리하는 기본적인 방법이다. 파일 시스템은 데이터를 파일이라는 단위로 저장하고, 디렉터리라는 구조를 통해 파일을 조직화 한다. 파일은 데이터를 담고 있는 기본 단위이며, 디렉터리는 이러한 파일을 그룹화하여 논리적인 구조를 제공한다. 현대의 파일 시스템은 다양한 기능을 제공하여 데이터의 저장, 검색, 삭제, 수정 등을 효율적으로 관리할 수 있게 한다.

파일 시스템은 물리적 저장 장치와 사용자가 데이터를 쉽게 관리할 수 있도록 하는 논리적 인터페이스 사이의 중개자 역할을 한다. 이는 데이터를 특정 형식으로 저장하고, 파일에 접근할 수 있는 방법을 제공하며, 파일의 메타데이터(예: 파일 크기, 생성 시간, 수정 시간 등)를 관리한다. 또한, 파일 시스템은 저장 공간을 효율적으로 사용하기 위해 공간 할당 및 해제를 관리하고, 데이터의 무결성을 보장하기 위한 다양한 메커니즘을 제공한다.

파일 시스템의 주요 구성 요소는 다음과 같다:

- 파일(File): 데이터를 저장하는 기본 단위이다. 각 파일은 이름과 확장자(예: .txt, .jpg, .exe 등)를 가지고 있으며, 파일 시스템은 파일의 위치와 크기, 접근 권한 등의 메타데이터를 관리한다.
- 디렉터리(Directory): 파일을 논리적으로 그룹화하는 구조이다. 디렉터리는 다른 디렉터리(하위 디렉터리)와 파일을 포함할 수 있으며, 파일 시스템의 계층적 구조를 형성한다.
- 블록(Block): 파일 시스템이 데이터를 저장하는 기본 단위이다. 각 파일은 하나 이상의 블록에 저장되며, 블록의 크기는 파일 시스템에 따라 다르다.
- 인덱스 노드(Inode): 파일에 대한 메타데이터를 저장하는 구조이다. 인덱스 노드는 파일의 크기, 소유자, 접근 권한, 데이터 블록의 위치 등을 포함한다.
- 저널링(Journaling): 파일 시스템의 무결성을 유지하기 위해 변경 사항을 기록하는 메커니즘이다. 저널링 파일 시스템은 데이터 변경을 저널에 먼저 기록한 후 실제 데이터를 수정하여 시스템 충돌 시 데이터 손실을 방지한다.

파일 시스템은 사용자와 컴퓨터 하드웨어 간의 인터페이스 역할을 한다. 사용자는 파일 시스템을 통해 데이터를 저장, 삭제, 검색, 수정할 수 있으며, 파일 시스템은 이러한 작업을 물리적 저장 장치에 반영한다. 예를 들어, 사용자가 파일을 생성하면 파일 시스템은 이를 위한 공간을 할당하고, 파일 메타데이터를 업데이트하며, 파일 데이터를 저장 장치에 기록한다.

2.2 파일 시스템의 중요성

파일 시스템은 운영 체제의 핵심 구성 요소 중 하나로, 사용자가 데이터를 저장하고 관리하는데 필수적이다. 파일 시스템은 데이터의 무결성을 보장하고, 데이터 손실을 방지하며, 데이터를 효율적으로 검색하고 접근할 수 있도록 한다. 파일 시스템이 없다면, 데이터의 체계적인 관리가 불가능하며, 이는 시스템의 성능 저하와 데이터 손실로 이어질 수 있다.

파일 시스템의 주요 역할은 다음과 같다:

- 데이터 조직화: 파일 시스템은 데이터를 파일과 디렉터리 구조로 조직화하여 사용자가 데이터를 쉽게 관리하고 접근할 수 있도록 한다. 이를 통해 데이터의 논리적 구조를 형성하고, 데이터 검색과 접근 속도를 향상시킨다.
- 데이터 무결성 보장: 파일 시스템은 데이터를 안정적으로 저장하고, 시스템 충돌이나 전원 차단 시 데이터 손상을 방지하기 위한 메커니즘을 제공한다. 예를 들어, 저널링 파일 시스템은 데이터 변경 사항을 저널에 기록하여 시스템 오류 시 데이터 손실을 최소화한다.
- 공간 관리: 파일 시스템은 저장 장치의 사용 가능한 공간을 효율적으로 관리하고, 파일과 디렉터리의 저장 위치를 최적화한다. 이를 통해 저장 공간의 낭비를 줄이고, 디스크 조각화(fragmentation)를 최소화한다.
- 접근 권한 관리: 파일 시스템은 파일과 디렉터리에 대한 접근 권한을 관리하여 사용자 간의 데이터 접근을 제어한다. 이를 통해 데이터 보안을 강화하고, 불법적인 접근을 방지할 수 있다.
- 성능 최적화: 파일 시스템은 데이터를 효율적으로 저장하고 검색할 수 있도록 다양한 최적화 기법을 제공한다. 예를 들어, 인덱스 구조를 사용하여 데이터 검색 속도를 향상시키고, 캐시 메커니즘을 통해 데이터 접근 속도를 높인다.

파일 시스템의 중요성은 데이터 관리의 효율성과 안정성에 있다. 데이터는 현대 사회에서 매우 중요한 자산이며, 이를 안전하게 저장하고 관리하는 것은 모든 컴퓨팅 시스템의 기본 요구사항이다. 파일 시스템은 이러한 요구사항을 충족시키기 위해 다양한 기능과 메커니즘을 제공하며, 이를 통해 사용자와 시스템의 데이터 관리 경험을 향상시킨다.

파일 시스템의 발전은 컴퓨터 과학의 중요한 부분을 차지한다. 초기 파일 시스템은 단순한 구조를 가지고 있었지만, 데이터 양이 증가하고 데이터 관리의 복잡성이 높아짐에 따라 더욱 복잡하고 효율적인 파일 시스템이 개발되었다. 예를 들어, FAT(File Allocation Table) 파일 시스템은 초기의 단순한 파일 시스템으로, 파일의 시작 위치와 다음 블록의 위치를 기록하여 파일을 관리하였다. 이후 NTFS(New Technology File System), ext3, ext4와 같은 파일 시스템은 저널링, 접근 권한 관리, 대용량 파일 지원 등 다양한 기능을 추가하여 파일 시스템의 효율성과 안정성을 크게 향상시켰다.

이처럼 파일 시스템은 컴퓨터 시스템의 중요한 구성 요소로서, 데이터의 효율적이고 안전한 관리에 필수적인 역할을 한다. 파일 시스템의 기능과 메커니즘을 이해하고 이를 효과적으로 사용하

는 것은 모든 컴퓨터 사용자와 개발자에게 중요한 기술이다.

2.3 파일 시스템 시뮬레이터의 종류와 특징

FAT (File Allocation Table)

1. FAT16 (File Allocation Table)

- 대부분의 Microsoft 운영체제에서 호환되며 단순한 구조이다.
- 최대 2GB까지만 지원한다.
- 암호화 및 압축이 불가능하다.
- 파일명 최대 길이는 영문 8자이다.
- 클러스터당 1,632KB를 할당하여 내부 단편화가 발생한다.

2. FAT32(File Allocation Table)

- FAT 16을 보강한 것으로, 최대 2TB까지 지원한다.
- 암호화 및 압축이 불가능하다.
- 파일명의 최대 길이는 영문 256자이다.
- 클러스터당 4KB 사용하여 내부 단편화를 줄였다.

NTFS (New Technology file System)

- 암호화 및 압축을 지원하며, 대용량 파일 시스템을 지원한다.
- 가변 클러스터 크기(512~ 64KB)이며, 기본 값은 4KB이다.
- 트랜잭션 로깅을 통한 복구/오류 수정이 가능하다.
- Windows NT 이상에서 지원한다.

EXT (Extended File System)

1. EXT(Extended File System)

- MINIX File System을 보완하여, 최대 2GB까지 파일 시스템 크기를 지원한다.
- 255byte까지 파일명을 지원한다.
- 접근 제어, inode 수정, 타임스탬프 수정 등의 기능이 불가능하다.
- 사용할수록 단편화가 심해진다.

2. EXT2(Second Extended File System)

- 파일 시스템은 최대 2GB까지 파일 시스템 크기를 지원되며, 서브 디렉터리 개수 제한이 대폭 증가하였다.
- FSCK를 사용한 파일 시스템 오류 수정을 지원한다.
- 캐시의 데이터를 디스크에 저장 중 오류 발생 시 파일 시스템에 손상이 올 수 있다.(Sync 이전 데이터 손실)
- FSCK 이용한 파일 복구 시간에 많은 시간이 소요된다.(전체 섹터 검사해야 됨)

3. EXT3(Third Extended File System)

- EXT2에 저널링 기능 추가 및 온라인 파일 시스템이 증대됐다.
- 파일 시스템 변경 시 저널에 먼저 수정 내용을 기록한다(갑작스러운 다운 시. 빠르게 오류 복구).

- 온라인 조각 모음이 불필요하다.(장시간 사용 시 조각화 발생).
- 디스크 조각화를 최소화한다.

4. EXT4(Fourth Extended File System)

- 16TB까지 파일 시스템을 지원하며, 볼륨은 1엑사바이트(Exabyte)까지 지원한다.
- Block Mapping 방식 및 Extends 방식을 지원한다.
- 저널 Checksum 기능이 추가되어 안정성이 강화되었다.
- 하위 호환성 지원 : ext3, ext2와 호환 가능
- Delayed allocation : 디스크에 쓰이기 전까지 블록 할당을 미루는 기술로 조각화 방지에 효과적
- 온라인 조각 모음 : 조각화 방지를 위한 커널 레벨의 기술
- Persistent pre-allocation : 파일 전체만큼의 공간은 사전 할당, 스트리밍, 데이터 베이스 등에 유용

UFS(Unix File System)

- VTOC 디스크 레이블 : 각 파티션의 기본 정보
- 부트블록 : 부트스트랩에 필요한 파일들
- 프라이머리 슈퍼블록 : 데이터 블록의 개수, 실린더 그룹의 개수, 마운트 정보
- 백업 슈퍼블록 : 각 실린더마다 슈퍼블록에 대한 복사본을 가짐
- 실린더 그룹 : 슈퍼블록, 실린더 그룹 블록, inode 테이블, 데이터 블록을 포함
- 슈퍼블록 : 파일 시스템 크기, i-node 테이블의 크기, free 블록 리스트 등 파일 시스템 관리 정보
- 실린더 그룹 블록 : 실린더 그룹 내의 유효 블록들의 비트맵 정보나 통계 정보
- i-node 테이블 : 파일에 대한 중요한 정보, 파일크기, 위치, 유형, 사용 허가권, 날짜 정보
- 데이터 블록 : 실제 데이터가 저장되는 공간

Btrfs (B-tree File System)

- 데이터 무결성 보장: 데이터와 메타데이터의 무결성을 확인하기 위해 체크섬을 사용한다.
- 스냅샷: 시점 복사(Copy-on-Write) 방식을 사용하여 데이터의 스냅샷을 빠르게 생성하고 관리할 수 있다.
- RAID 지원: 소프트웨어 RAID 기능을 제공하여 데이터 중복과 오류 복구를 지원한다.
- 압축: 데이터 압축 기능을 통해 저장 공간을 절약할 수 있다.
- 중복 제거: 중복 데이터를 제거하여 저장 공간을 효율적으로 사용할 수 있다.
- 서브볼륨: 하나의 파일 시스템 내에서 여러 서브볼륨을 생성하여 관리할 수 있다.
- 온라인 파일 시스템 디스크 검사 및 복구: 파일 시스템을 마운트한 상태에서 디스크 검사 및 복구 작업을 수행할 수 있다.
- 확장성: 파일 시스템의 크기를 온라인 상태에서 동적으로 조정할 수 있다.

ZFS (Zettabyte File System)

- 데이터 무결성 보장: 엔드-투-엔드 데이터 무결성을 제공하며, 모든 데이터와 메타데이터에 대해 체크섬을 사용한다.

- 스냅샷 및 복제: 효율적인 스냅샷과 복제 기능을 제공하여 데이터의 시점 복사와 백업을 용이하게 한다.
- RAID-Z: 기존 RAID보다 향상된 소프트웨어 RAID 기능을 제공하여 데이터 보호와 성능을 최적화한다.
- 압축: 데이터 압축 기능을 통해 저장 공간을 절약할 수 있다.
- 중복 제거: 데이터 중복 제거 기능을 제공하여 저장 공간의 효율성을 높인다.
- 풀(pool) 기반 스토리지 관리: 여러 디스크를 하나의 풀로 결합하여 스토리지 자원을 유연하게 관리할 수 있다.
- 자체 치유(self-healing): 데이터 오류를 자동으로 감지하고 복구하는 기능을 제공한다.
- 확장성: 페타바이트 규모의 대용량 스토리지를 효율적으로 관리할 수 있다.

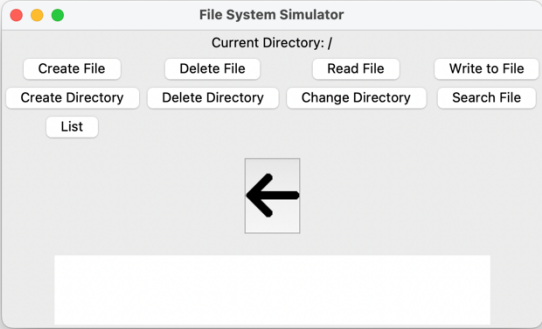
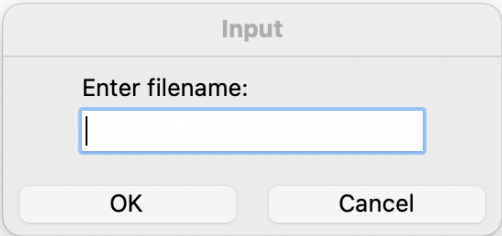
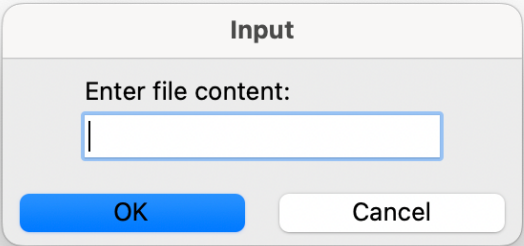
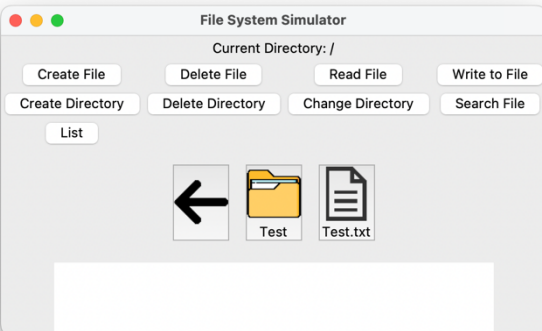
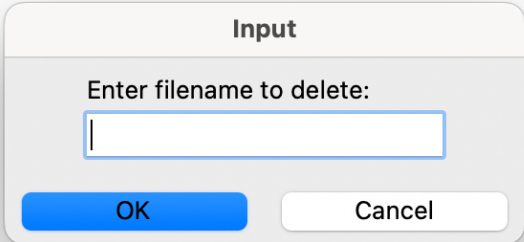
3. Program description

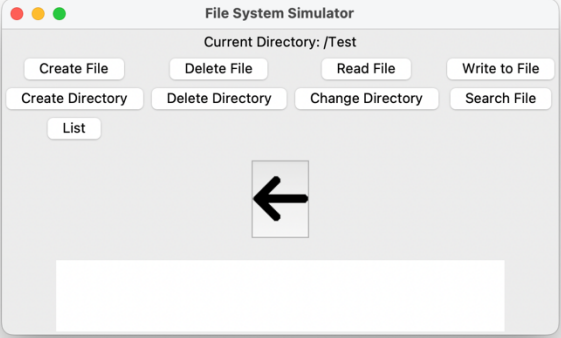
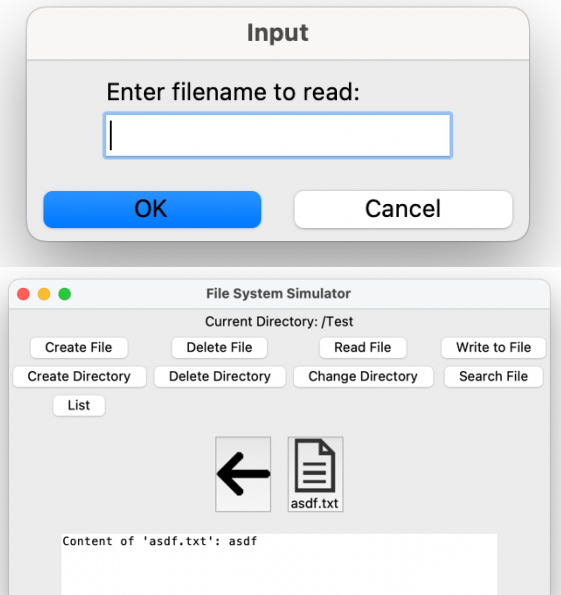
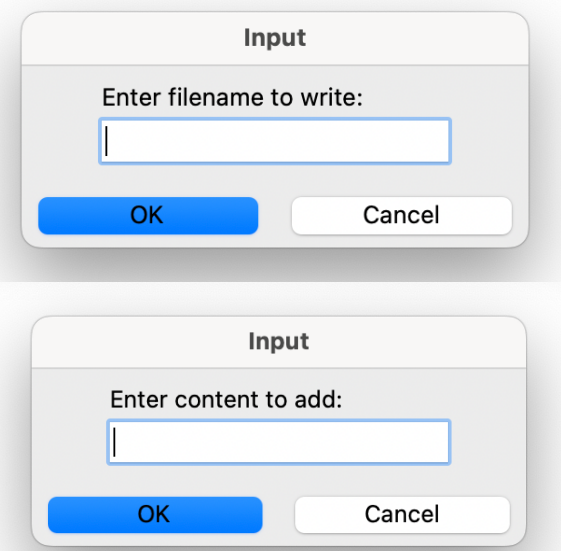
FileSystemSimulatorGUI 클래스는 파일 시스템 시뮬레이터의 그래픽 사용자 인터페이스(GUI)를 관리하고 사용자와의 상호작용을 처리하는 역할을 한다. 이 클래스는 Tkinter를 사용하여 창, 라벨, 버튼, 텍스트 박스 등을 초기화하고 구성하며, 파일 및 디렉터리 생성, 삭제, 읽기, 쓰기, 디렉터리 변경, 파일 검색 등의 작업을 위해 사용자로부터 입력을 받는다. 또한, 사용자 입력에 따라 FileSystemSimulator 클래스의 메서드를 호출하여 파일 시스템 작업을 수행하며, 작업 결과나 오류 메시지를 출력 텍스트 박스에 표시하여 사용자에게 피드백을 제공한다. 마지막으로, 현재 디렉터리의 파일 및 디렉터리 목록을 시각적으로 표시하여 사용자가 쉽게 탐색할 수 있도록 한다.

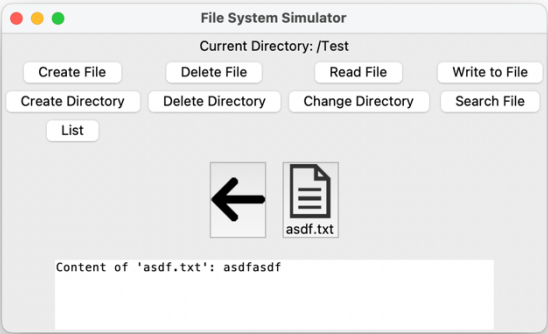
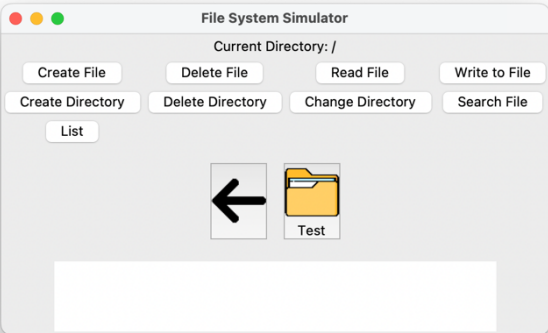
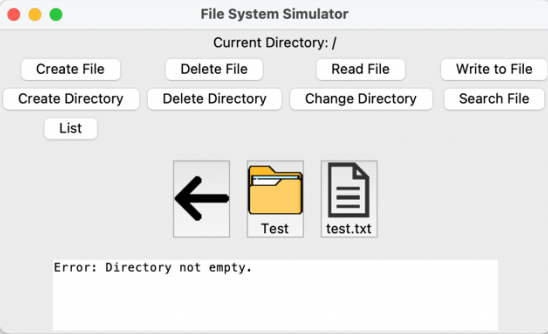
FileSystemSimulator 클래스는 파일 시스템의 논리적 구조를 관리하고, 파일 및 디렉터리와 관련된 작업을 수행하는 역할을 한다. 이 클래스는 지정된 이름과 내용을 가진 파일 또는 디렉터리를 생성하고, 지정된 파일 또는 디렉터리를 삭제하며, 파일의 내용을 읽고 새로운 내용을 파일에 추가하거나 덮어쓴다. 또한, 현재 작업 중인 디렉터리를 변경하고, 파일 시스템 전체에서 지정된 파일을 검색하여 경로를 반환하며, 현재 디렉터리의 파일 및 디렉터리 목록을 제공한다. 작업 중 발생한 오류는 콜-백 함수를 통해 처리하여 사용자에게 알린다.

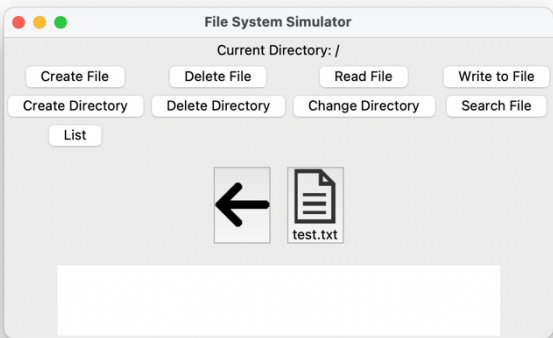
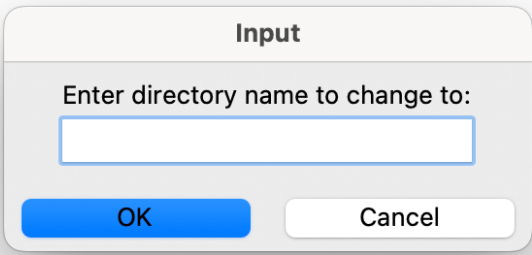
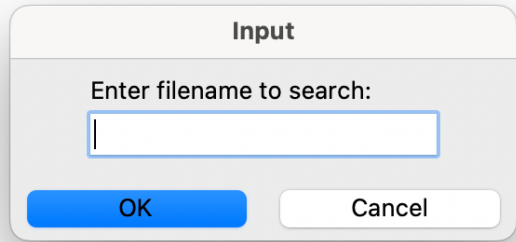
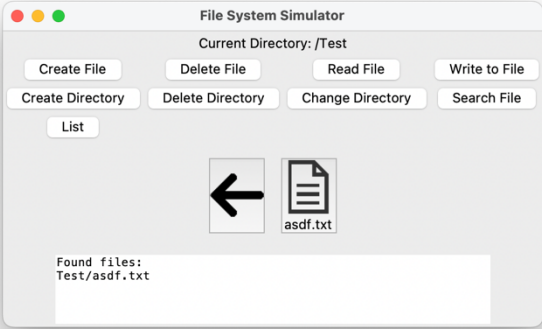
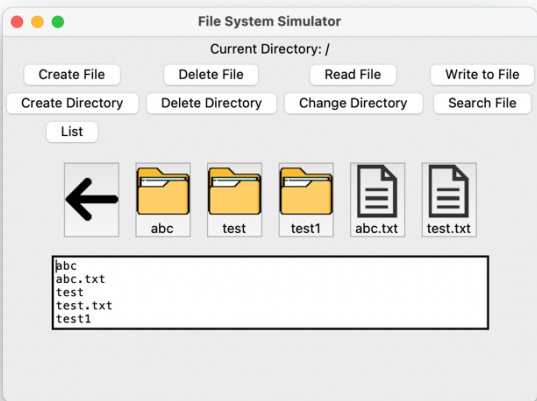
이 두 클래스는 서로 협력하여 파일 시스템 시뮬레이터의 기능을 완성한다. FileSystemSimulatorGUI 클래스는 사용자와의 상호작용을 처리하고, FileSystemSimulator 클래스는 실제 파일 시스템 작업을 수행함으로써, 사용자에게 직관적이고 시각적인 방식으로 파일 시스템의 기본 작업을 수행하고 학습할 수 있는 환경을 제공한다.

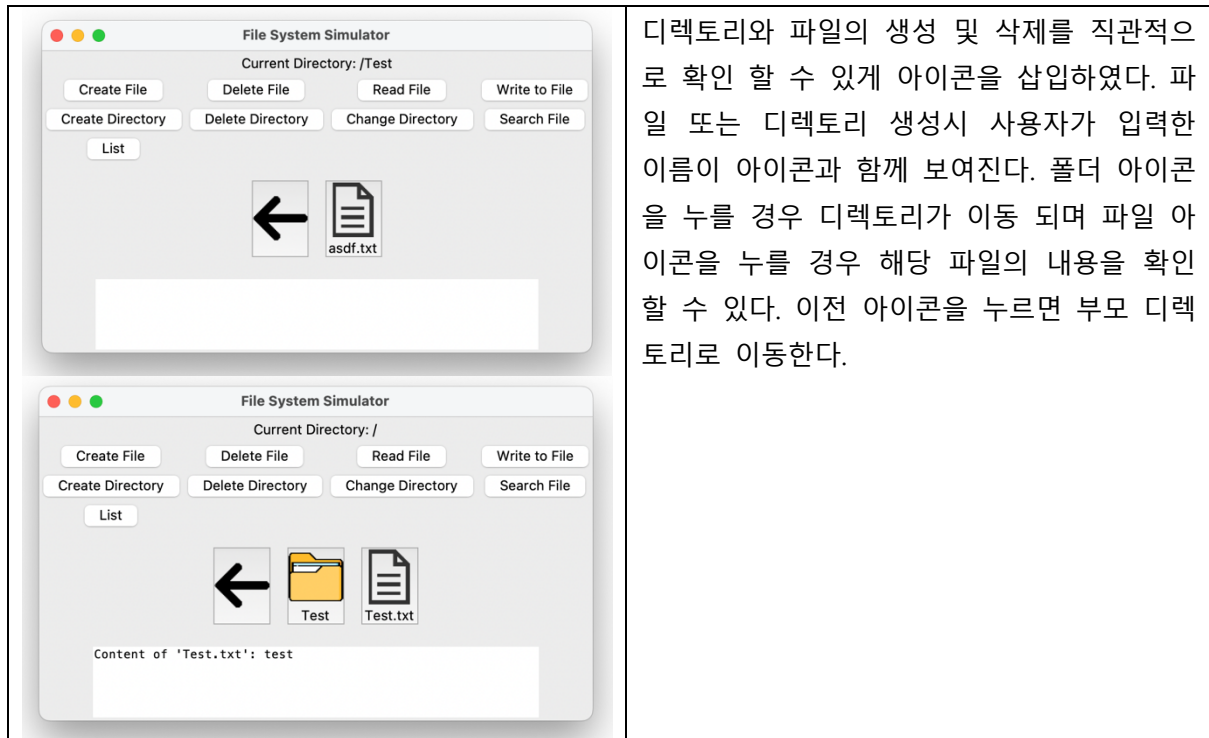
4. Result

 The image shows the main window of the 'File System Simulator'. It has a title bar with three colored buttons (red, yellow, green). Below the title bar, it says 'Current Directory: /'. There are two rows of buttons: the first row contains 'Create File', 'Delete File', 'Read File', and 'Write to File'; the second row contains 'Create Directory', 'Delete Directory', 'Change Directory', and 'Search File'. Below these buttons is a 'List' button. In the center of the window is a large black arrow pointing left. At the bottom is a large white rectangular text box.	<p>프로그램을 실행하면 다음과 같이 시뮬레이터가 실행된다. 사용자는 본 화면에서 원하는 작업을 시작할 수 있다.</p>
   The first image is an 'Input' dialog box with the title 'Input'. It contains the text 'Enter filename:' followed by a text input field. At the bottom are 'OK' and 'Cancel' buttons. The second image is another 'Input' dialog box with the title 'Input'. It contains the text 'Enter file content:' followed by a text input field. At the bottom are 'OK' and 'Cancel' buttons. The third image shows the 'File System Simulator' main window after creating a file. It now displays a folder icon labeled 'Test' and a file icon labeled 'Test.txt' next to the left-pointing arrow. The text box at the bottom is still empty.	<p>Create File 버튼을 누르면 다음과 같은 창이 생성된다. 파일 이름을 입력하고 파일의 내용을 입력하면 초기화면에 입력한 파일명을 가진 파일 아이콘이 생성된 것을 확인 할 수 있다.</p>
 The image shows an 'Input' dialog box with the title 'Input'. It contains the text 'Enter filename to delete:' followed by a text input field. At the bottom are 'OK' and 'Cancel' buttons.	<p>Delete File 버튼을 누르면 다음과 같은 창이 생성된다. 파일명을 입력하면 파일이 삭제되고 아웃풋 텍스트 박스가 초기화 된다.</p>

	
	<p>Read File 버튼을 누르면 다음과 같은 창이 생성된다. 파일명을 입력하면 아웃풋 텍스트 박스에 해당 파일의 내용이 보여진다. 현재 디렉토리 내에 파일이 없다면 에러가 발생하며 파일을 찾을 수 없음을 알린다.</p>
	<p>Write to File 버튼을 누르면 다음과 같은 창이 생성된다. 파일명과 내용을 입력하면 해당 파일의 기존 내용에 새로 입력한 내용이 추가된다.</p>

	
<div data-bbox="236 631 767 878"> <p>Input</p> <p>Enter directory name to create:</p> <input type="text"/> <p>OK Cancel</p> </div> <div data-bbox="225 934 775 1265">  </div>	<p>Create Directory 버튼을 누르면 다음과 같은 창이 생성된다. 디렉토리 명을 입력하면 입력한 디렉토리 이름을 가진 폴더 아이콘이 생성된 것을 확인할 수 있다.</p>
<div data-bbox="236 1319 767 1565"> <p>Input</p> <p>Enter directory name to delete:</p> <input type="text"/> <p>OK Cancel</p> </div> <div data-bbox="225 1619 775 1951">  </div>	<p>Delete Directory 버튼을 클릭하면 다음과 같은 창이 생성된다. 삭제할 디렉토리 명을 입력했을 때 해당 디렉토리가 비어 있지 않다면 에러가 발생하며 디렉토리가 삭제되지 않는다. 디렉토리가 비어 있는 상태라면 삭제되며 폴더 아이콘이 사라지고 아웃풋 텍스트 박스가 초기화 된다.</p>

	
	<p>Change Directory 버튼을 누르면 다음과 같은 창이 생성된다. 이동할 디렉토리 명을 입력하면 해당 디렉토리로 이동할 수 있다. 부모 디렉토리로의 이동은 '..'을 입력해야 한다.</p>
 	<p>Search File 버튼을 누르면 다음과 같은 창이 생성된다. 찾고 싶은 파일의 이름을 입력하면 파일의 경로가 아웃풋 텍스트 박스에 나타난다.</p>
	<p>List 버튼을 누르면 아웃풋 텍스트 박스에 해당 디렉토리에 있는 모든 디렉토리와 파일들이 나열된다. 아웃풋 텍스트 박스의 크기가 제한되어 있기 때문에 디렉토리와 파일의 총 개수가 5개 이상이 되면 한 번에 보여지지 않고 스크롤 하여 확인 할 수 있다.</p>



디렉토리와 파일의 생성 및 삭제를 직관적으로 확인 할 수 있게 아이콘을 삽입하였다. 파일 또는 디렉토리 생성시 사용자가 입력한 이름이 아이콘과 함께 보여진다. 폴더 아이콘을 누를 경우 디렉토리가 이동 되며 파일 아이콘을 누를 경우 해당 파일의 내용을 확인할 수 있다. 이전 아이콘을 누르면 부모 디렉토리로 이동한다.

5. Conclusion

이번 프로젝트를 통해 개발된 파일 시스템 시뮬레이터는 파일 및 디렉토리 생성, 삭제, 읽기, 쓰기, 검색, 디렉토리 변경 등 기본적인 파일 시스템 작업을 시각적으로 이해할 수 있도록 돕는 유용한 도구이다. FileSystemSimulator 클래스는 파일 시스템의 논리적 작업을 처리하고, FileSystemSimulatorGUI 클래스는 사용자와의 상호작용을 관리하여 직관적인 사용자 인터페이스를 제공한다. 이 두 클래스는 상호작용하여 사용자에게 파일 시스템의 기본 개념과 동작 원리를 학습할 수 있는 환경을 제공한다.

결론적으로, 파일 시스템 시뮬레이터는 파일 시스템의 기본 개념과 동작을 이해하는 데 필요한 기능을 효과적으로 제공한다. 이 시뮬레이터를 통해 사용자는 파일 시스템의 기본 원리를 쉽게 학습하고, 다양한 파일 시스템 작업을 실습할 수 있다.