**운영체제 가을과제**

**Client-Server 모델을 이용한 Login Application 구현**

201221092 이상윤

2016.11.16

**목차**

1. 개요
2. 프로그램 구조설계
3. 기능 리스트
4. 동작시험 결과
5. 프로젝트 진행보고
6. 프로그램 계획
7. 계획 대비 구현정도
8. 향후 개선사항
9. **개요**

이 프로그램은 클라이언트 서버 구조를 기반으로 pipe를 통하여 프로세스 간 통신을 수행하는 기능을 구현하는 것을 목적으로 한다.

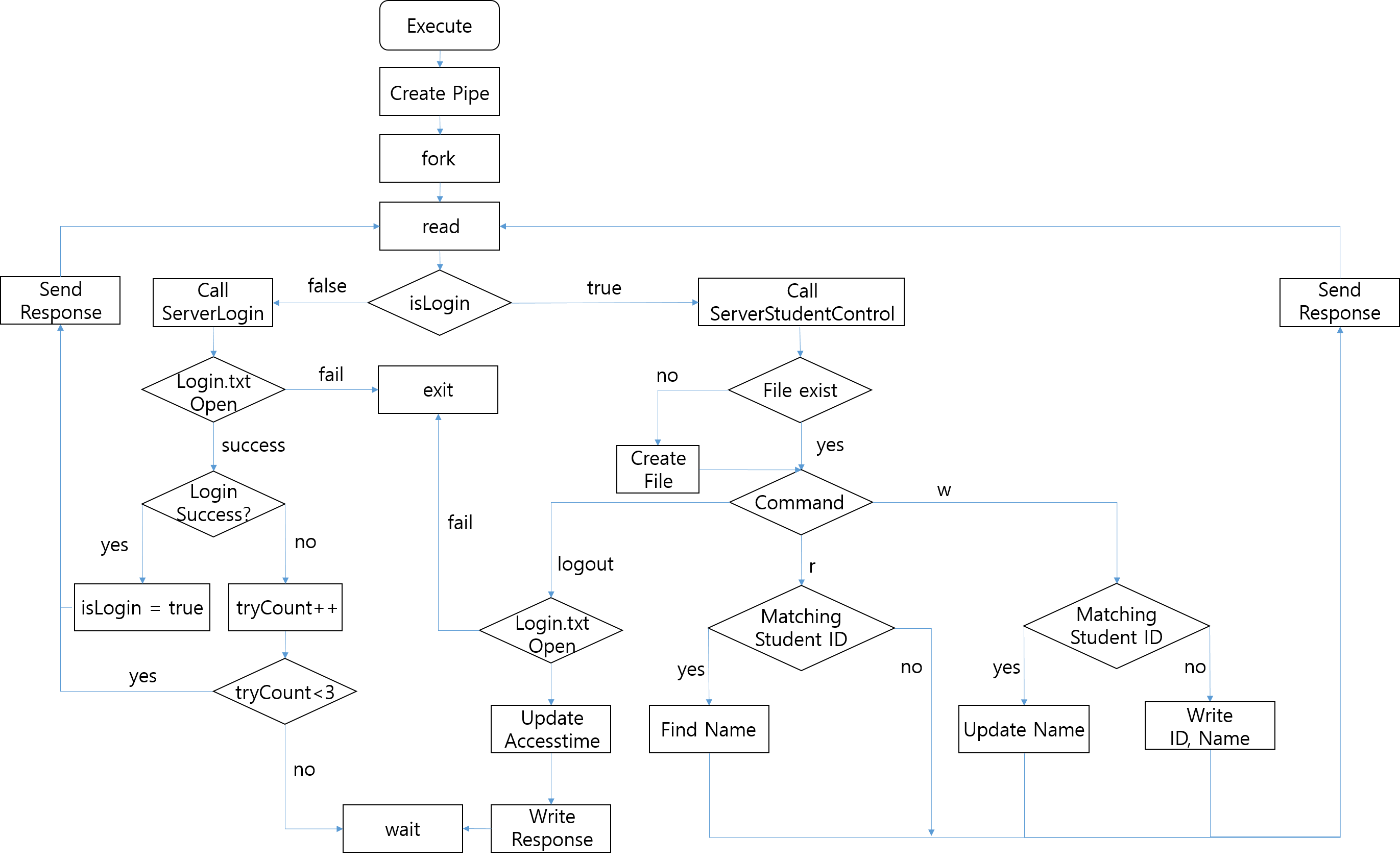
프로그램이 실행되면 서버(부모 프로세스)에서 pipe를 생성하고 클라이언트(자식 프로세스)를 fork한다. 프로세스 생성 후에 수행되는 기능은 크게 다음과 같이 나눌 수 있다.

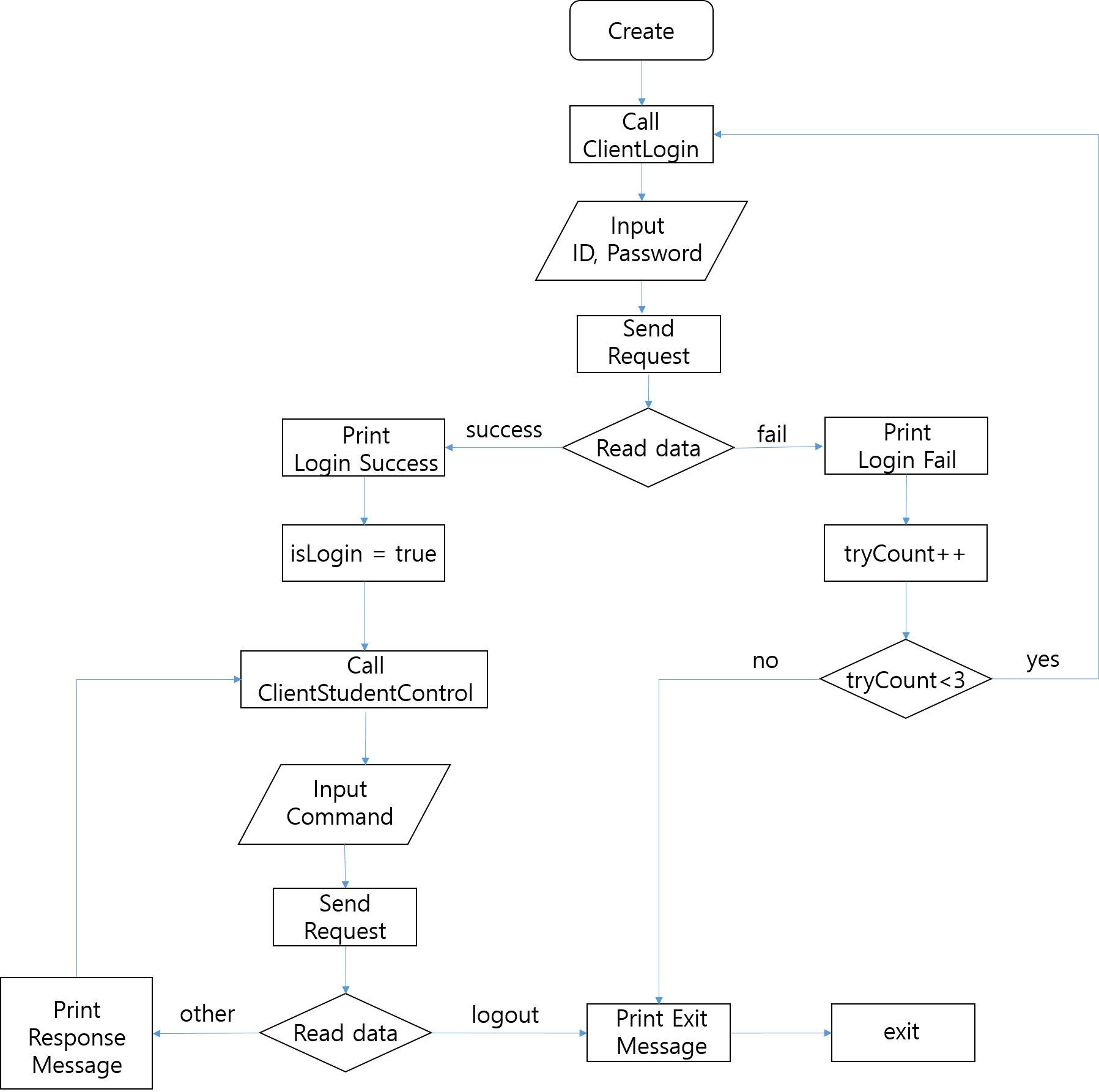
1. 클라이언트는 로그인 명령어와 함께 로그인 정보를 입력받아 서버로 pipe를 통해 전송한다. 서버는 데이터를 전송받으면 login.txt 라는 파일에 접근하여 해당 사용자의 로그인 아이디, 패스워드, 최종 접근시간 데이터와 비교하여 유효한 데이터인지 판단하여 클라이언트에게 pipe를 통해 전송한다.

이 때, login.txt 파일에 접근한 유저의 최종 접근시간이 업데이트 되고, 클라이언트는 전송받은 정보에 따라 화면에 적절하게 출력한다. 로그인 정보가 유효하지 않다면 사용자는 로그인 요청을 세 번까지 시도할 수 있으며, 로그인에 성공했을 경우 다음 명령어를 수행 가능하고, 로그인 요청이 세 번 실패했을 경우는 더이상 명령어를 입력할 수 없다.

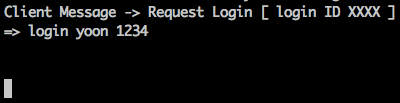
2. 로그인 후 클라이언트는 pipe를 통하여 서버에게 파일 접근 요청을 보낼 수 있고, 서버는 클라이언트로부터 요청이 오면 파일 접근을 수행한다. 서버는 요청의 종류( r 학번, w 학번 이름 ) 에 따라 해당 학번에 대한 이름을 검색, 학번 및 이름 추가/수정 과 같은 작업을 수행하고 성공/실패 정보를 클라이언트에게 전송한다. 클라이언트는 서버에서 전송 받은 데이터를 적절하게 화면에 출력한다.

3. 로그인 후 클라이언트가 로그아웃 명령을 하면 클라이언트(자식) 프로세스를 종료하고 서버는 해당 사용자 정보를 login.txt 파일에서 찾아 최종시간을 업데이트 한 후 wait 상태가 된다.

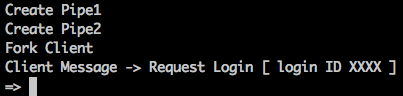
1. **프로그램 구조설계 – 서버, 클라이언트 프로세스로 나누어 설계**
2. **Server**
3. **Client**

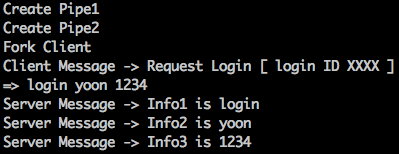


|  |  |  |
| --- | --- | --- |
| **3. 기능 리스트** | | |
| **기능** | **상세설명** | **테스트 번호** |
| **1. Pipe 생성** | 1. 책과 실습예제 코드를 참고하여 구현 | #1, #2 |
| **2. Fork 수행** | 1. fork() 시스템 함수를 통해 자식 프로세스 생성 후childpid 변수에 자식 프로세스의 pid 저장(예제) |
| **3. 명령어 분석** | 1. 클라이언트에서 명령어 입력을 통해 서버에게 요청을 하게 되면 서버는 dataParsing 함수를 호출하여 전송 받은 명령어를 분석하여 info 변수에 각각 저장 | #3 |
| **4. 로그인 요청** | 1. 클라이언트는 ClientLogin 함수를 호출하여 사용자로부터 입력을 받고, 서버로 전송.  2. 서버는 명령어 정보를 분석하여 로그인 요청이라면 Bool 타입 ServerLogin 함수를 호출하고 login.txt 파일 접근  파일이 존재하지 않는다면, 서버 메세지를 화면에 출력하고 프로그램 종료. | #4, #5 |
| **5. 로그인 성공** | 1. login.txt 파일에 저장된 데이터를 fscanf 함수로 읽어 요청한 로그인 정보와 일치하는지 비교하고, 최종 접근시간보다 현재 시간이 이전이라면 클라이언트에게 로그인 성공 메세지를 전송  2. 파일에 해당되는 정보의 위치 커서를 조정하여 최종 접근시간 업데이트  3. 서버에서 로그인 여부를 판단하는 변수인 isLogin 변수에 ServerLogin 함수의 리턴값으로 true 할당  4. 클라이언트는 성공 메세지를 화면에 출력  5. 클라이언트에서 로그인 여부를 판단하는 변수인 isLogin 변수를 true 로 변경 | #6, #7 |
| **6. 로그인 실패** | 1. login.txt 파일을 끝까지 읽을 때 까지 일치하는 데이터가 없다면 클라이언트에게 로그인 실패 메세지를 전송 2. 서버에서 로그인 여부를 판단하는 변수인 isLogin 변수에 ServerLogin 함수의 리턴값으로 false 할당   3. 클라이언트는 실패 메세지를 화면에 출력  4. 클라이언트와 서버에서 각각 로그인 시도 횟수를 저장하는 tryCount 변수의 값을 1 증가 | #8, #9 |
| **7. 로그인 3회 실패** | 1. tryCount 변수가 3 이하라면 로그인 요청을 계속 시도할 수 있고, 로그인 실패로 tryCount의 값이 3이 되면 화면에 메세지를 출력 후 wait | #10 |
| **8. 접근시간 갱신** | 1. time 라이브러리의 현재시간 정보를 담고 있는 구조체를 timeToString 함수로 문자열로 login.txt 파일에 갱신 2. 최종접근시간 정보가 쓰여있는 위치를 찾는 과정은 파일에서 일치하는 정보가 있을 때 break 로 탐색을 멈추고 현재 커서를 시간 정보 문자열의 길이만큼 앞으로 이동하여 다시 쓰는 방식으로 구현 | #11 |
| **9. 접근시간 비교** | 1. 로그인 요청 시 login.txt 에 저장된 최종 접근시간 문자열과, 현재 시간정보 문자열을 파라미터로 사용하는 timeCompare 함수를 호출해 년, 월, 일, 시, 분, 초 값에 각각의 단위 (12, 31, 24, 60, 60) 를 곱하여 나오는 상수 값을 비교하는 알고리즘 사용 | #12 |
| **10. 추가 명령** | 1. 클라이언트는 로그인 성공(isLogin = true) 이 된 경우 추가 명령을 입력받고 전송하는 ClientStudentControl 함수를 호출 2. 로그인이 된 상태에서 클라이언트로 부터 데이터를 전송받으면 ServerStudentControl 함수를 호출 | #13 |
| **11. 파일 읽기 요청** | 1. 클라이언트가 r 명령어와 학번을 입력하여 서버로 전송 2. 서버는 정보를 분석하여 r 요청을 처리 3. 서버는 students.txt 파일에 저장된 학번과 요청한 학번을 비교하여 일치하는 정보가 있다면 해당 학번에 대한 이름을 클라이언트로 전송하고 화면에 메세지 출력 4. 일치하는 정보가 없다면 읽기 실패 메세지를 전송하고 화면에 메세지 출력 | #14, #15 |
| **12. 파일 쓰기 요청** | 1. 클라이언트가 w 명령어와 학번, 이름을 입력하여 전송 2. 서버는 정보를 분석하여 w 요청을 처리 3. students.txt 파일에 이미 저장된 학번 정보가 없다면 해당 학번과 이름 정보 파일에 저장 후 클라이언트에게 쓰기 성공 메세지 전송 4. 클라이언트는 화면에 메세지 출력 | #16 |
| **13. 파일 수정 요청** | 1. 클라이언트가 w 명령어와 학번, 이름을 입력하여 전송 2. 서버는 정보를 분석하여 w 요청을 처리 3. students.txt 파일에 이미 저장된 학번 정보가 있다면 해당 학번에 대한 이름 정보를 수정 후 클라이언트에게 수정 성공 메세지 전송 4. 클라이언트는 화면에 메세지 출력 | #17 |
| **14. 로그아웃** | 1. 로그인 성공(isLogin = true) 후 클라이언트에서 logout 명령어를 서버에 전송하면 마찬가지로 서버는 명령어를 분석하여 로그아웃 요청 처리 2. 서버는 login.txt 파일을 읽고, 현재 로그인 되어 있는 사용자의 세션정보가 담긴 위치를 찾아 최종 접근시간 업데이트 후 클라이언트에게 로그아웃 완료 데이터 전송 3. 서버와 클라이언트에서 화면에 메세지를 출력후 wait | #18, #19 |
| **15. 잘못된 명령어** | 1. 로그인 시 login 또는 LOGIN 외 다른 명령어 입력 시 2. 로그인 후 r, w, logout 요청 외 다른 명령어 입력 시 3. 서버에서 잘못된 요청이라는 메세지를 클라이언트에게 전송하고 클라이언트는 화면에 메세지 출력 | #20, #21 |

**4. 동작 시험 결과**

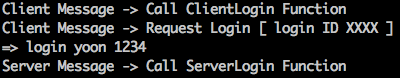
#1 – 처음 실습 예제를 실행하는 과정에서 교수님 말씀처럼 쉽게 실행 되지 않았다. 파이프 번호를 잘못 입력하여 로그인 요청 시 파이프로 데이터를 전송하는 부분에서 문제가 발생했다.



#2 – 파이프 생성과 자식 프로세스 fork 가 정상적으로 수행되는지 확인하기 위해 프린트를 해보았고, 정상적으로 잘 실행되는 것을 확인했다.

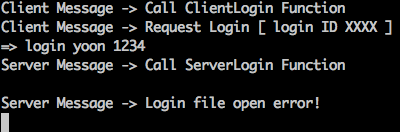
#3 – 가장 먼저 클라이언트에서 요청하는 로그인 부분에서 데이터가 정상적으로 넘어가는지 판단하기 위해 전송받은 데이터를 파싱하는 함수를 만들어 가공한 다음 프린트 해보았다.

C언어를 배운지 오래되어 문자열을 자르는 함수가 기억나지 않아 구글에서 검색하여 사용했고, 그 결과 명령어 이름, 파라미터 값이 잘 출력되었다.

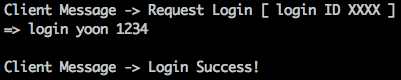




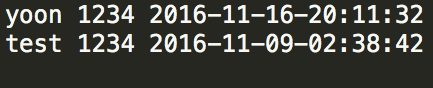
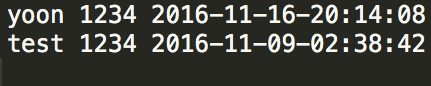
#4 – 클라이언트는 로그인 요청을 하기 위해 ClientLogin 이라는 함수를 호출하고 사용자의 입력을 파이프를 통해 서버로 전송하게 되면 서버는 ServerLogin 이라는 함수를 호출한다.

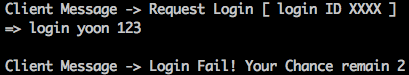


#5 - login.txt 파일을 열 때 r+ 모드로 열었는데, 아직 login.txt 파일을 생성하지 않아 에러가 발생하였다. 바로 login.txt 파일을 생성하고 다시 시도해보았다.

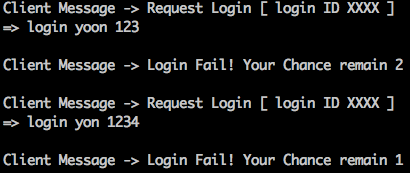


#6 – 로그인이 성공하면 성공했다는 메시지를 출력한다. 로그인에 성공하고 나서야 클라이언트와 서버의 파이프 통신방식을 조금 이해한 느낌이었다. 데이터를 보내고 받는 것을 자유롭게 할 수 있으니 남은 기능들도 잘 구현할 수 있을 것 같았다.

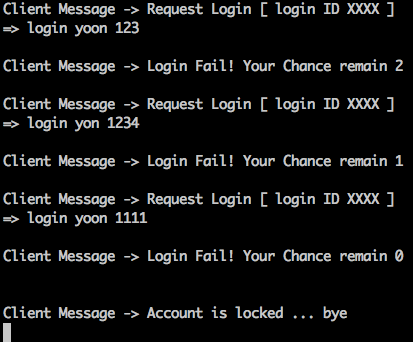
#7 – 로그인 성공 후 login.txt 파일의 최종접근시간 정보가 업데이트 된 것을 확인하였다.



#8 – 로그인이 실패했을 경우 화면에 메시지를 출력하고, 남은 시도 횟수를 같이 출력하여 사용자에게 주의를 주었다. ( login.txt 파일에 저장된 아이디와 비밀번호는 yoon 1234 이다. )



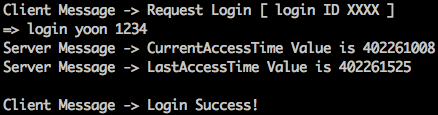
#9 – 한번 더 로그인 실패를 했을 경우 tryCount 를 증가시켜 화면에 출력해주었다.



#10 – 로그인 실패를 3번 해서 남은 시도 횟수가 0이 되면 게정이 잠겼다는 메시지를 출력하고 추가 명령을 수행할 수 없게 하였다.



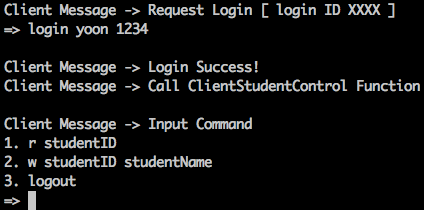
#11 – 로그인 시 최종접근시간의 업데이트는 일치하는 정보가 있을 때 파일탐색을 멈추고 현재 커서의 위치를 최종접근시간 문자열의 길이인 20 만큼 앞으로 이동시킨다음 현재 시간정보가 담긴 구조체 t를 timeToString 함수를 이용하여 문자열로 변환 후 fprintf 함수를 사용하여 파일에 덮어쓰는 방식을 사용했다..

#12 – 로그인 시 ‘최종접근시간이 현재시간보다 이전이라면’ 라는 조건이 들어가있었는데, 문자열로 저장된 시간정보와 시간 구조체를 어떻게 비교할지 막막했다. 두 가지 방법중 고민을 했는데,

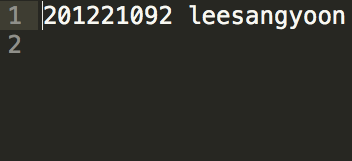
첫번째는 문자열정보를 파싱하여 시간구조체를 생성한 다음 현재시간 구조체와 비교하는 방법이었고, 두번째는 반대로 현재시간 구조체를 이미 만들어둔 timeToString 함수를 이용해 문자열로 변환하여 문자열끼리 비교하는 방법이었다.

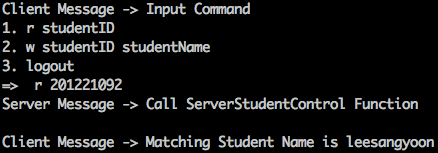
고민을 하다가 두번째 방법을 선택하였는데 그 이유는 첫번째 방법을 사용하면 아주 간단하게 시간구조체를 비교하는 내장함수를 사용할 수 있었지만 문자열로 시간을 비교하는 방법을 한번 사용해보고 싶었다. 또, 이미 구현해둔 timeToString 함수를 재사용할 수 있어서 비교 알고리즘 외 추가 코드가 생기지 않기 때문이었다.

문자열 두개를 파라미터로 받는 timeCompare 함수를 선언하고, 문자열에서 각각 년, 월, 일, 시, 분, 초 정보를 추출했다. 그 다음 각각의 단위인 12, 31, 24, 60, 60 을 곱하여 나오는 수를 출력해 보았다.

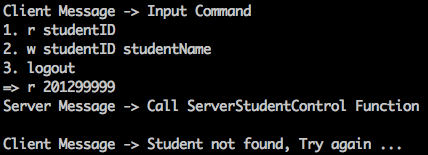
그 결과 시간정보를 담고 있는 문자열을 잘 비교할 수 있다는 것을 확인하고 로그인 조건에 추가하였다.

#13 – 로그인에 성공하면 클라이언트에서 ClientStudentControl 함수를 호출하여 추가 명령어를 입력할 수 있게 화면에 출력해주었다.

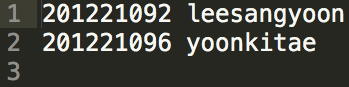


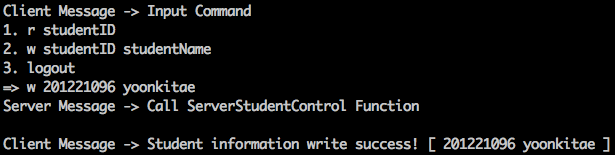


#14 – students.txt 파일에 다음과 같이 정보가 입력되어 있을 때, 먼저 r 명령을 통해 학번정보를 서버에 전송해보았다. 그 결과 서버에서 ServerStudentControl 함수를 호출하여 r 요청을 처리하였다. 먼저 students.txt 파일을 열어 한 줄씩 정보를 읽어 일치하는 학번이 있는지 비교하고 만약 있다면 해당 이름을 찾아 클라이언트에게 전송하였다.

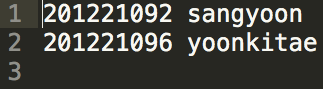
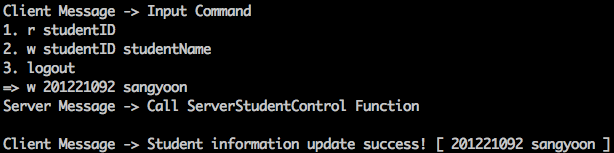
클라이언트는 전송받은 데이터를 화면에 출력해주었다.

#15 – 만약 일치하는 학번이 없다면 서버는 클라이언트에게 일치하는 정보가 없다는 메시지를 전송하고 클라이언트는 화면에 메시지를 출력한다.



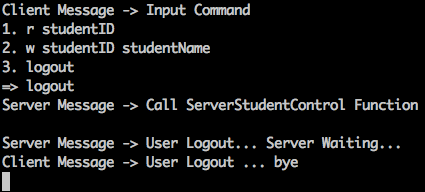
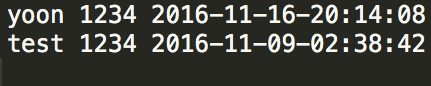
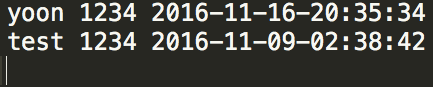
#16 – 다음으로 w 명령과 친구의 학번, 이름을 서버에 전송해보았다.

마찬가지로 서버는 ServerStudentControl 함수를 호출하여 w 요청을 처리하였다. 먼저 파일에서 일치하는 학번이 있는지 검사하여 일치하는 학번이 없기 때문에 새로 파일에 학번, 이름 정보를 써주었다.

서버는 클라이언트에게 write success 라는 메시지와 함께 추가된 정보를 전송하고, 클라이언트는 화면에 메시지를 출력한다.

#17 – 이미 존재하는 학번이 있을 경우 업데이트를 하기 위해 w 요청을 다시 해보았다.

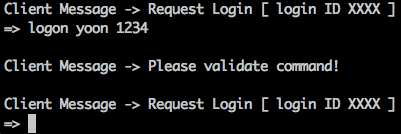
그 결과 서버에서 일치하는 학번을 찾아 이름 정보를 파일에 업데이트 해준 것을 확인했고, 서버는 클라이언트에게 update success 라는 메시지와 함께 업데이트된 정보를 전송하고, 클라이언트는 화면에 메시지를 출력한다.

#18 – 마지막으로 로그아웃 요청을 서버에 전송하였다. 마찬가지로 서버는 ServerStudentControl 함수를 호출하여 logout 요청을 처리하였다. 서버는 클라이언트에게 로그아웃이 되었다고 메시지를 보내고 wait 상태가 된다. 클라이언트는 logout이 되었다는 메시지를 받으면 화면에 출력한다.

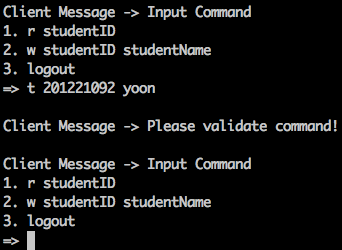
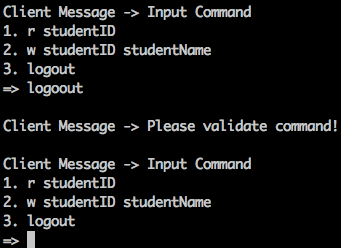
#19 – 로그아웃 시 최종접근시간을 갱신하는 부분이 처음에 막막했다. 로그인을 할 때는 사용자의 아이디와 비밀번호가 있기 때문에 login 파일에서 위치를 쉽게 찾을 수 있었다. 하지만 로그아웃 요청시에는 logout 이라는 명령어만 보내기 때문에 어떻게 현재 사용자의 정보가 있는 위치를 찾아야 할지 몰랐다.

그래서 이전에 로그인한 정보를 다른 변수에 담아두면 로그아웃 할 때 사용할 수 있지 않을까라는 생각을 했다. 로그인 세션의 역할을 하는 변수를 만들고 로그인 시 정보를 담아놓았다.

그리고 나서 로그아웃 시 login.txt 파일에서 해당 위치를 찾아 최종접근시간을 업데이트 하는 방식으로 처리하였다.



#20 – 요구하는 기능을 모두 구현하고 나서 잘못된 입력에 대한 처리를 해보았다.

로그인 시 login 또는 LOGIN 명령어가 아니라면 잘못된 명령어라는 메시지를 출력해주었다.

#21 – 로그인 후 추가 명령어를 입력할 때 r, w, logout 이외의 다른 명령어를 입력하면 잘못된 명령어라는 메시지를 화면에 출력해주었다.

**5. 프로젝트 진행보고**

1. 프로그램 계획

프로젝트를 시작하기 전에 파이프 통신 예제를 먼저 실행해보기로 했다. 수업시간에 교수님께서 예제 프로그램을 실행하는 것도 쉽지 않을 것 이라고 하셔서 예제 코드를 작성한 다음 실행해 보았다. 다음으로 클라이언트와 서버간 데이터를 주고받는 부분을 구현해보았다. 데이터를 잘 주고받을 수 있으면 어떤 기능이라도 구현할 수 있을 것이라고 생각했다. 데이터를 주고받는 것을 구현하고 나서 기능별로 함수를 나누어 프로그램 로직에 맞게 구현해 나가기로 하였다. 유효하지 않은 입력에 대한 처리는 모든 기능을 구현한 뒤에 처리하기로 계획했다.

1. 계획 대비 구현정도

파이프 생성과 자식 프로세스 fork -> 완료

서버와 클라이언트 간 데이터 전송 -> 완료

클라이언트와 서버에서 로그인 함수를 따로 분리해서 처리 -> 완료

로그인 실패 시 3번까지 시도하는 로직 구현 -> 완료

로그인 후 추가 명령을 처리하는 함수를 따로 분리해서 처리 -> 완료

학생정보 읽기 기능 -> 완료

학생정보 쓰기, 수정 기능 -> 완료

로그인, 로그아웃 시 최종접근시간 업데이트 기능 -> 완료

로그아웃, 로그인 시도 3번 실패 시 서버 wait -> 완료

잘못된 명령어 입력 처리 -> 완료

명령어에 따른 잘못된 정보 입력 -> 완벽하지 않음

전체적으로 server(), client() 함수에서는 프로그램의 흐름만 쉽게 알아볼 수 있도록 기능은 함수로 나누어서 처리하였다.

각각의 함수 내에서도 다른 역할을 하는 코드는 따로 분리해서 함수로 정의하였다.

( timeToString, timeCompare, dataParsing 함수 )

1. 향후 개선사항

* 로그인 후 추가 명령을 처리하는 함수 내에서 r, w, logout 을 처리하는 부분도 따로 함수로 나누어 처리하면 코드의 가독성이 더 좋아질 것이다.
* 명령어에 따라 추가로 입력하는 정보를 잘못 입력했을 경우 ( fscanf 에서 형식에 맞지 않게 입력, fgets 에서 공백을 포함하여 입력 ) 적절한 메시지를 출력해 준다면 사용자가 프로그램을 바르게 사용할 수 있도록 할 수 있을 것이다.