**System Programming Project 1**

담당 교수 : 김영재

이름 : 이제현

학번 : 20170624

1. **개발 목표**

* **해당 프로젝트에서 구현할 내용을 간략히 서술.**
* **(MyShell을 만드는 전체적인 개요에 대해서 작성하면 됨.)**

Bash shell과 비슷하게 MyShell을 만들어 각종 명령어들을 수행하는 나만의 쉘 스크립트를 만든다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* **아래 항목을 구현했을 때의 결과를 간략히 서술**

1. Phase 1

먼저 Phase 1을 구현했을때 ./myshell을 통해 명령어를 받을 준비를 하고 ls, cd, cat 등의 명령어를 받고 명령어에 맞게 동작한다. 쉘 스크립트는 quit 또는 exit을 받으면 종료되고 interrupt인 Ctrl+C를 입력받게 되면 프로세스가 종료한다. (단 쉘은 종료하지 않는다.) 또한 명령어 뒤에 &를 붙여 background로 프로세스를 수행할 수 있도록 만든다. 명령어들이 문법에 맞지 않는 경우 오류 표시를 적절히 수행한다.

1. Phase 2

Phase 2를 구현하면 파이프라인 기능을 사용할 수 있다. “|”를 기준으로 왼쪽의 output이 오른쪽에 input으로 들어간다. |를 여러개 사용하는 다중 파이프도 구현할 수 있다. 이것은 process 끼리 통신할수 있도록 만드는 것이다.

1. Phase 3
   1. **개발 내용**

* **아래 항목의 내용만 서술**
* **(기타 내용은 서술하지 않아도 됨. 코드 복사 붙여 넣기 금지)**
* **Phase1 (fork & signal)**
  + fork를 통해서 child process를 생성하는 부분에 대해서 설명

pid\_t인 pid에 fork를 통해 child process를 만든다. (pid = fork()) 만일 여기서 pid가 -1이면 에러를 표시하고 pid가 0인 부분부터 내용을 작성한다.

* + connection을 종료할 때 parent process에게 signal을 보내는 signal handling하는 방법 & flow

waitpid 함수를 통해 자식 프로세스가 죽을 때 까지 기다리도록 만든다. 만일 -1이라면 오류를 출력한다.

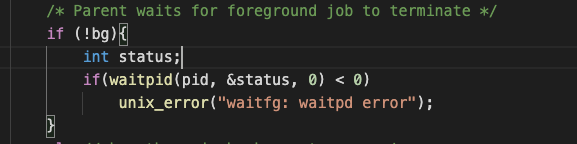
* **Phase2 (pipelining)**
  + Pipeline( ‘|’ )을 구현한 부분에 대해서 간략히 설명 (design & implementation)

명령어를 판단하는 eval 함수와 별개로 파이프라인을 발견했을 때 판단하는 함수를 만든다. 이는 함수 void eval\_p(char \*cmdline);이다.

* + Pipeline 개수에 따라 어떻게 handling했는지에 대한 설명

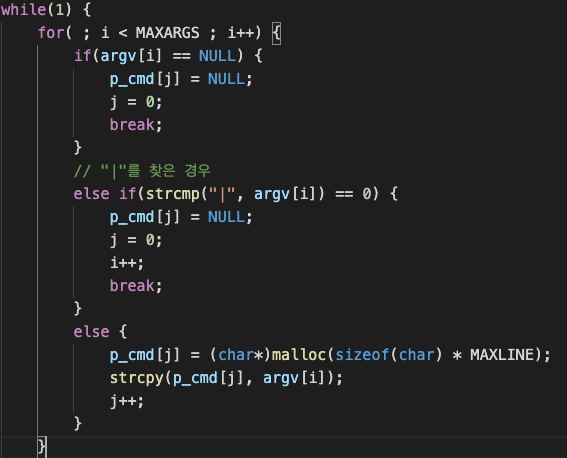
여러개의 Pipeline을 받을 수 있도록 Pipeline을 찾고 명령어를 구분짓는 부분에서, Pipeline을 찾는 반복적인 방식으로 구현했다. File descriptor를 2개를 생성해서 하나의 pipeline 이 끝날때 기존의 값들을 두번째 fd에 저장해놓고 이를 사용하는 방식으로 구현했다.

* **Phase3 (background process)**
  + Background (’&’) process를 구현한 부분에 대해서 간략히 설명
  1. **개발 방법**
* **B.의 개발 내용을 구현하기 위해 어느 소스코드에 어떤 요소를 추가 또는 수정할 것인지 설명. (함수, 구조체 등의 구현이나 수정을 서술)**
* **Phase 1**

void eval(char \*cmdline) 함수 부분이다. 먼저 fork를 하여 자식 프로세스를 생성했다. 여기서 signal 함수들을 통해 INTR을 주는 부분을 구현했고 strcat으로 char \*root에 명령어를 붙였다. 그리고 밑의 if(execve(root,argv,environ) <0) 으로 해당 명령어를 실행과 동시에 명령어 오류 판단을 하도록 했다.

if(waitpid(pid, &status, 0) < 0)를 통해 error detection과 wait를 동시에 구현한다.

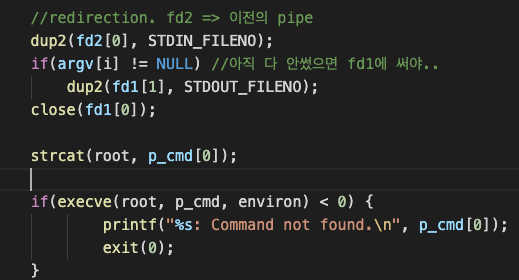
error를 출력할때 내장된 함수 unix\_error를 이용하였다.

- **Phase 2**

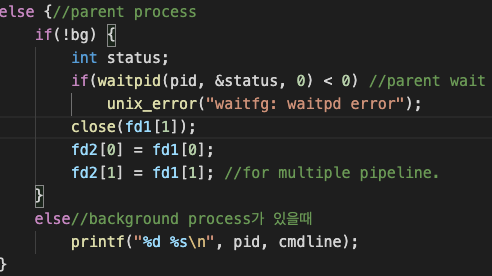
먼저 이것은 eval\_p라는 pipeline인 경우 ( “|”를 발견했을 때) 문자열을 자르는 부분이다.

이 함수를 통해 argv를 순방하며 pipeline을 찾고 명령어를 구분지어 p\_cmd에 명령어를 넣어준다.

while문으로 반복되어 여러개의 pipeline을 찾을 수 있다.



해당 부분은 Pipe의 input과 output을 설정해주는 dup2를 사용한 부분이다. 설명은 위와 같다.

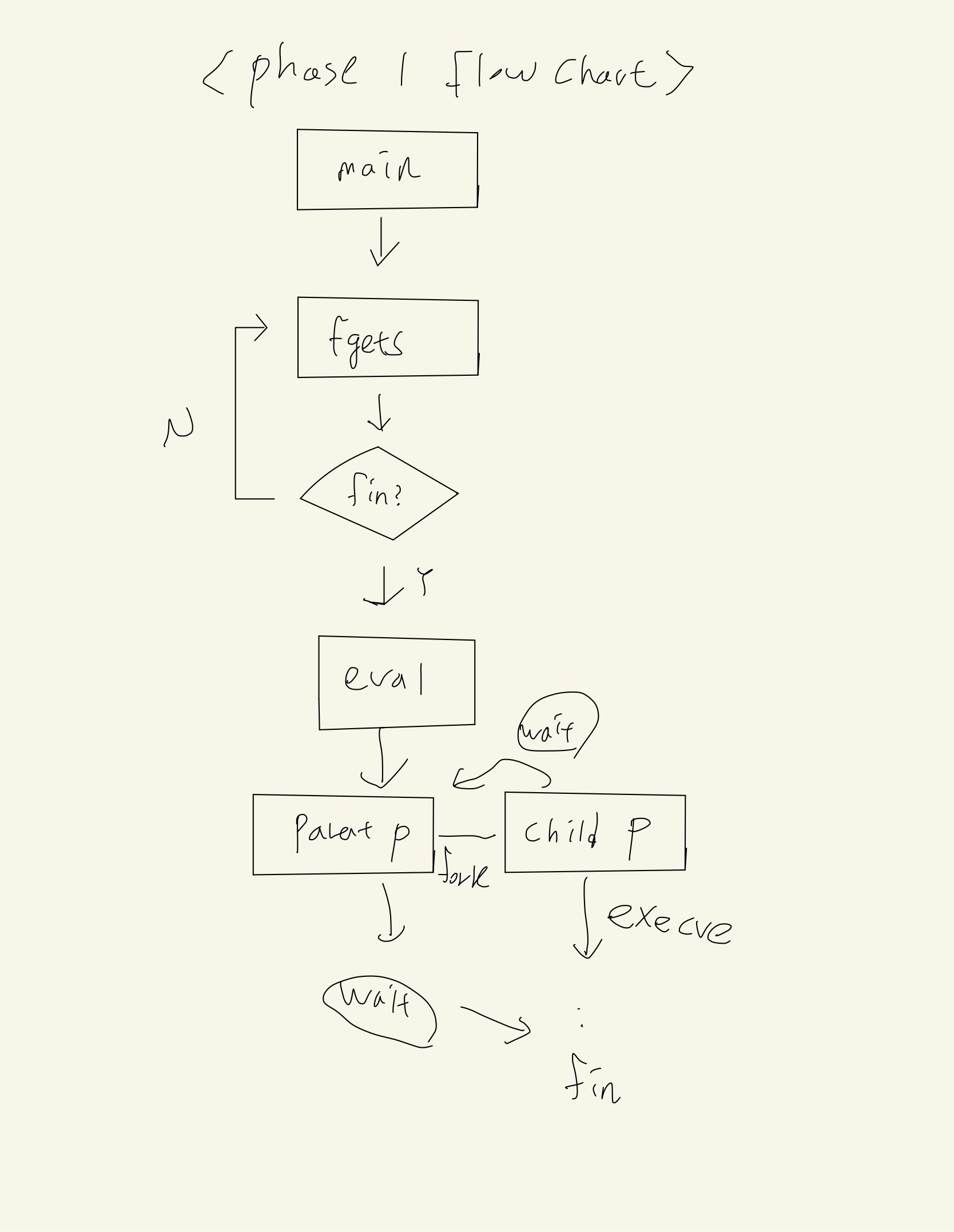
Phase 1과 유사하게 root = “/bin/“에 pipeline이 제거된 명령어 p\_cmd를 붙여 이를 실행한다.****

그리고 fd2에 fd1을 저장하는데, 저장된 fd2는 이후에 while loop를 돌며 파이프라인이 여러개인 경우 활용된다. 위의 dup2(fd2[0], STDIN\_FILENO)가 이전의 output이 input으로 들어오는 부분이다.

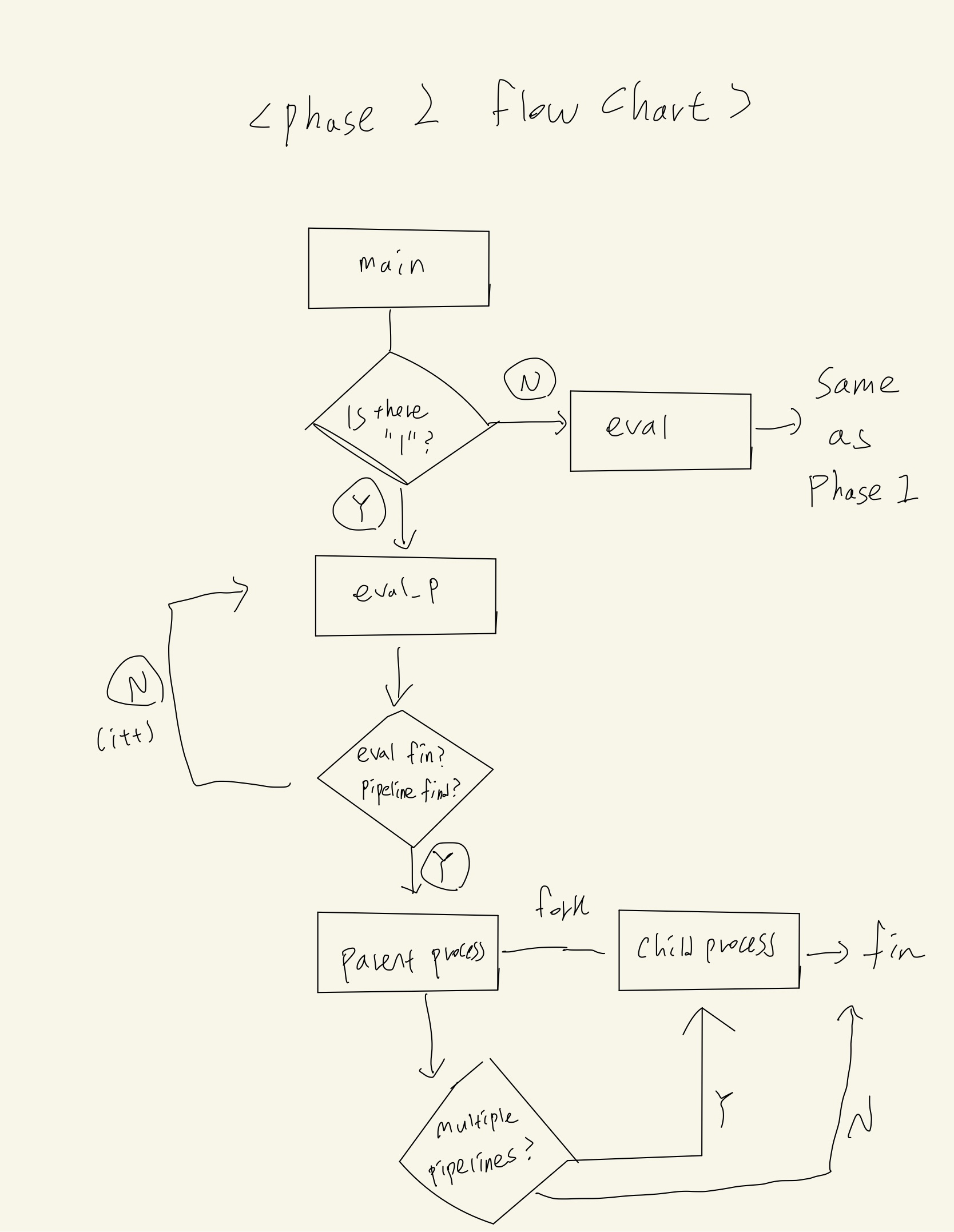
1. **구현 결과**
   1. **Flow Chart**

* **2.B.개발 내용에 대한 Flow Chart를 작성.**
* **(각각의 방법들에서 추가된 내용(fork, pipeline, background)만 특성이 잘 드러나게 그리면 됨.)**

1. **Phase 1 (fork)**

****

1. **Phase 2 (pipeline)**

****

1. **Phase 3 (background)**