Assignment #1: Finding Sequence

System Programming

202111322 오색빛

제출일: 2022.10.30.

Design

- 프로그램 실행 시, 찾아야할 substring(str)과 fork할 프로세스의 개수(i)를 인자로 전달하면, 미리 지정된 input 배열에서 모든 substring을 찾아 각 sequence의 시작 index를 모두 출력한다.

- 자식 프로세스를 i개 만들어 각 프로세스가 input 배열을 비슷한 양으로 나눠 병렬처리한다.

- overlap area에 해당하는 부분에 있는 sequence도 찾아 출력할 수 있도록 처리한다.

- 각 자식 프로세스는 탐색하면서 시퀀스를 찾으면 pipe에 시작 index를 write한다.

- 각 자식 프로세스는 자신에게 할당된 개수를 탐색하면 종료한다.

- 부모 프로세스는 자식 프로세스가 모두 종료하면 pipe에서 모두 read한다.

- pipe로 읽은 sequence의 시작 index를 정렬하여 출력한다.

Implementation

1. Valuable Description

|  |  |  |
| --- | --- | --- |
| Type | Identifier | Description |
| const char\* | str | 찾을 substring (argv[1]) |
| const int | pcount | 자식 프로세스의 개수 (argv[2]) |
| pid\_t \* | pid | 자식 프로세스 pid 저장 배열 |
| int \* | N | 각 프로세스가 탐색해야할 문자의 개수 배열 |
| int \* | pipefd | pipe file descriptors  (pipefd[0]: read end, pipefd[1]: write end) |
| int \* | foundIndex | 부모 프로세스에서 자식 프로세스들이  찾은 sequence의 시작 index를 저장하는 배열 |
| int | count | 탐색의 시작부분을 계산하기 위한 변수 |
| int | buf | pipe에서 read할 때 사용되는 버퍼 |
| int | fcount | 찾은 sequence의 개수 |
| int | min | 선택정렬에 사용되는 변수, 최솟값 index |

2. Code Description

(1) Main 함수

텍스트이(가) 표시된 사진

자동 생성된 설명

main함수로 들어온 인자가 3개가 아닌 경우, 찾으려는 문자열이 없는 경우(NULL), 병렬 프로세스의 개수가 0보다 작거나 같은 경우 종료하도록 예외처리 했다.

파이프가 생성되지 않은 경우에도 종료하도록 했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

각 프로세스마다 비슷한 양의 일을 처리하도록 처리할 개수를 담는 배열을 생성했다.

각 프로세스는 우선 input배열의 크기를 프로세스 개수로 나눈 몫을 나눠 갖고, 만약 나머지가 존재한다면 앞에서부터 1씩 나눠 갖도록 구현했다.

ex) MAXS = 10, pcount = 3 🡺 N[] = { 4, 3, 3 }

텍스트이(가) 표시된 사진

자동 생성된 설명

부모 프로세스는 pcount 개수만큼의 자식 프로세스를 생성한다.(fork)

fork 전 count값을 갱신해서 각 자식 프로세스가 처리해야할 범위를 계산한다.

자식 프로세스는 쓰기 기능만 하면 되므로 사용되지 않는 read end(pipefd[0])은 close한다.

자식 프로세스는 findStr 함수를 통해 input에서 substring sequence를 찾아 pipe에 write한다.

처리해야할 범위를 끝내면 write end(pipefd[1])를 close하고 종료한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

부모 프로세스는 읽기 기능만 하면 되므로 사용하지 않는 write end(pipefd[1])은 close한다.

자식 프로세스가 모두 종료되면 pipe에 있는 값을 모두 read하여 foundIndex 배열에 저장한다.

더 이상 읽을 값이 없을 경우 read end(pipefd[0])를 close한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

선택 정렬을 이용해서 foundIndex를 오름차순으로 정렬했다.

배열을 순차 탐색하면서 최솟값을 찾아 앞부분과 자리를 바꾸어 정렬한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

정렬된 foundIndex를 출력한다.

(2) findStr 함수 (str: 찾을 문자열, s: 시작 위치, n: 탐색 문자 개수, fd: pipe file descriptor)

텍스트이(가) 표시된 사진

자동 생성된 설명

(바깥 for문) 각 자식 프로세스는 input[s]부터 input[end-1]까지 탐색한다. found에 검색을 시작하는 index(k)를 저장한다. (안쪽 for문) input[i]가str[j]과 같을 경우, j와 i값을 증가하면서 str의 글자수만큼 반복을 진행한다. 같지 않은 부분이 나타날 경우, found값을 -1로 바꾸고 break한다. 안쪽 for문이 끝날 때 found가 -1이 아니라면, 즉 substring을 찾았다면 found값(sequence 시작 index)을 pipe에 write한다.

문자를 검사할 때 i 변수를 이용하여, i는 end보다 큰 것이 가능하므로 overlap area부분도 검사할 수 있도록 처리했다. 바깥 for문이 n번 반복되면서 문자열이 겹치더라도 올바른 결과값을 찾아낸다. ex) ABABABA 에서 ABA 찾기 🡺 (0, 2, 4)

Function description

|  |  |  |
| --- | --- | --- |
| Function name | Arguments | Description |
| main | int argc | 메인함수에 전달되는 정보의 개수 |
| char\* argv[] | 프로그램을 실행할 때 메인 함수에 전달되는 인자의 문자열들이 저장되는 배열, argv[0]은 프로그램 실행경로로 고정 |
| Return Value: int | 정상적인 종료: 0, 비정상적인 종료: -1 |
| findStr | const char\* str | 찾으려고 하는 문자열 |
| int s | input배열에서 문자열을 찾기 시작하는 위치 index |
| int n | process에 할당된 탐색할 문자의 개수 |
| int fd | write할 pipe file descriptor (write end) |
| Return Value: void |  |