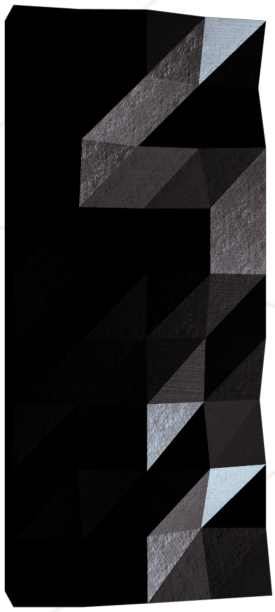

Get Next Line

Reading a line on a fd is way too tedious

Summary: *이 프로젝트의 목적은 file descriptor로부터 읽혀진, newline으로 끝나는 라인을 반환하는 함수를 코드화 하는 것입니다.*

* 이 파일은 프로젝트 수행을 좀 더 빠르고 편하게 하기 위해 만들어진 파일입니다.
* 개인적으로 보기위해 만들어진 파일이므로 오역이 있을 수 있습니다.
* 영어버전과 함께 보시는 걸 추천 드립니다.

Contents

Chapter I

Goals

이 프로젝트는 너의 collection에 매우 편리한 함수를 추가하는 것을 허용하지 않을 것입니다. 하지만 또한 C 프로그래밍에서 매우 흥미로운 새 개념을 배울 수 있을 것입니다.

Chapter II

Common Instructions

- 프로젝트는 Norm 규칙에 맞춰 작성되어야 합니다. 보너스 파일/함수가 있는 경우,해당 파일/함수들은 norm 검사에 포함되며, norm error가 있을 시, 0점을 받게 될것입니다.
- 함수들은 정의되지 않은 행동들과는 별개로 예기치 않게 중단되어서는 안 됩니다.(예를 들어, segmentation fault, bus error, double free 등.)
- 필요한 경우 heap에 할당된 모든 메모리 공간은 적절하게 해제되어야 합니다. 메모리 누수는 용납되지 않을 것입니다.
- 그 과제가 필요하다면, Makefile을 제출해야 합니다. 그것은 -Wall -Wextra -Werror 플래그를 지정하여 컴파일할 것입니다. 그리고 Makefile은 relink 되어서는안됩니다.
- Makefile은 최소한 $(NAME), all, clean, fclean, re를 포함해야 합니다.
- 프로젝트에 보너스를 제출하려면, Makefile에 보너스 규칙을 포함해야 합니다. 이보너스 규칙은 프로젝트의 메인 부분에서 금지되었던 모든 다양한 헤더, 라이브러리,또는 함수들은 추가해야 할 것입니다. 보너스는 반드시 _bonus.{c/h}라는 다른 파일에 있어야 합니다. 의무적으로 해야 될 파트과 보너스 파트는 별도로 평가될 것입니다.
- 프로젝트에서 당신의 libft를 허용한다면, 소스들과 그것과 연관된 Makefile을 연관된 Makefile과 함께 libft폴더에 복사해야 합니다. 프로젝트의 Makefile은 Makefile을 사용하여 라이브러리를 컴파일한 다음, 프로젝트를 컴파일해야 합니다.
- 이 과제물을 제출하지 않고 등급이 매겨지지 않을지라도, 우리는 당신의 프로젝트를위한 테스트 프로그램을 만들 것을 권장합니다. 그것은 너의 work와 peer's work를쉽게 테스트할 기회를 제공할 것입니다. 너는 defence하는 동안 이 테스트 프로그램들이 특히 유용하다는 것을 알게 될 것입니다. 사실, defence하는 동안, 너는 너의 테스트 프로그램과 평가 받는 동료의 테스트 프로그램들을 자유롭게 사용할 수 있을것입니다.
- 할당된 git 저장소에 과제물을 제출하세요. 오직 git 저장소에 있는 과제물만 등급이매겨질 것입니다. 만약 너의 과제를 평가받는데 Deepthought가 배정된다면, 그것은동료평가 이후에 이루어질 것입니다. 만약 Deepthought 평가 중에 오류가 발생한다면, 그 즉시 평가는 중지될 것입니다.

Chapter III

Mandatory part - Get_next_line

Function name	`get_next_line`
Prototype	`int get_next_line(int fd, char **line);`
Turn in files	`get_next_line.c, get_next_line_utils.c, get_next_line.h`
Parameters	`#1. file descriptor for reading` `#2. The value of what has been read`

Return value	`1 : A line has been read`
	`0 : EOF has been reached`
	`-1 : An error happened`

External functs.	`read, malloc, free`
Description	`Write a function which returns a line read from a file descriptor, without the newline.`

- get_next_line 함수를 loop 안에서 호출하면 file descriptor에서 사용할 수 있는 텍스트를 EOF가 올 때까지 한 번에 한 줄씩 읽을 수 있을 것입니다.
- 파일에서 읽을 때, 표준입력으로부터 읽어드릴 때 함수가 제대로 동작하는지 확인하십시오.
- libft는 이 프로젝트에서 사용할 수 없습니다. get_next_line이 동작하는 데 필요한 함수들이 들어있는 get_next_line_utils.c 파일을 추가해야 합니다.
- 당신의 프로그램은 -D BUFFER_SIZE=xx 플래그를 붙여서 컴파일 해야 합니다. 그것은 너의 get_next_line에서 read함수를 호출하기 위한 buffer size로 사용될 것입니다.
- 컴파일을 이런 식으로 진행됩니다 : gcc -Wall -Wextra -Werror -D BUFFER_SIZE=32 get_next_line.c get_next_line_utils.c
- 너의 read 함수는 표준입력으로 또는 파일로부터 읽어드리기 위해서 컴파일할 때 정의된 BUFFER_SIZE를 사용해야 합니다.
- get_next_line.h (헤더 파일)에는 적어도 get_next_line 함수의 프로토타입이 있어야 합니다.

BUFFER_SIZE 값이 9999인 경우에도 함수는 여전히 작동하나요? BUFFER_SIZE 값이 1이라면? 10000000이라면? 왜 그런지 아시나요?

get_next_line이 호출될 때마다 가능한 한 적게 읽도록 해야 합니다. 만약 newline을 만나면, 현재 line을 반환해야 합니다. 전체 파일을 읽지 말고 각 line을 처리하세요.

테스트하지 않고 너의 프로젝트를 제출하지 마세요. Cover your bases. 실행할 많은 테스트들이 있습니다. file로부터, redirection으로부터, stdin으로부터의 읽기를 시도하세요. 표준 출력에 newline을 보낼 때 프로그램은 어떻게 동작하나요? , CTRL-D?

- 만약 동일한 file descriptor의 두 호출 사이에서 첫 번째 fd에서 EOF에 도달하기 전에 다른 파일로 전환될 경우, 우리는 get_next_line이 정의되지 않은 동작을 가진다고 생각합니다.
- lseek은 허용된 함수가 아닙니다. 파일 읽기는 오로지 한번만 행해져야합니다.
- 마지막으로 바이너리 파일을 읽을 때, 우리는 get_next_line이 정의되지 않은 동작을 가진다고 생각합니다. 그러나 당신이 원한다면, 이러한 동작을 논리적으로 만들 수 있습니다.
- 전역 변수는 금지되어 있습니다.

좋은 시작은 정적 변수가 무엇인지 아는 것일 겁니다 :
https://en.wikipedia.org/wiki/Static_variable

Chapter IV

Bonus part

get_next_line 프로젝트는 간단해서 보너스를 받기 위해 할 수 있는 것들이 거의 없지만, 나는 당신이 상상력이 풍부하다고 확신합니다. 만약 당신이 mandatory part를 완벽하게 숙달했다면, 어떻게든 더 나아가 이 보너스 파트를 완성하세요. 다시 말하지만, mandatory part가 완벽하지 않다면, 보너스는 고려되지 않을 것입니다.
이 파트를 위한 all 3 initial files with _bonus를 제출하세요.

- Single static variable로 get_next_line 성공하는 것.
- get_next_line을 사용하여 다중 file descriptor를 관리 할 수 있는 것. 예를 들어, file descriptor 3, 4, 5가 읽을 수 있는 경우, get_next_line은 3에서 한 번, 4에서 한 번, 다시 3에서 한 번, 5에서 한 번 호출할 수 있습니다. 각 descriptor의 reading thread를 잃지 않고..