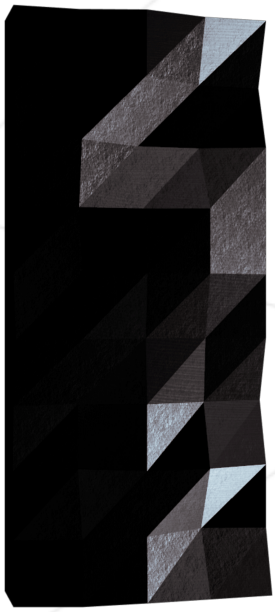

Get Next Line

fd-ից մեկ տող կարդալը չափից դուրս ձանձրալի է:

Հակիրճ. այս նախագծի նպատակը մի այնպիսի ֆունկցիա կոդավորելն է, որը կվերադարձնի ֆայլի նկարագրիչից կարդացած տող:

Ցանկ

Գլուխ I

Նպատակներ

Այս նախագծի շնորհիվ ոչ միայն մի նոր ֆունկցիա կսովորեք, այլև նոր գիտելիք ձեռք կբերեք C-ի մասին: Խոսքը ստատիկ փոփոխականների մասին է:

Գլուխ II

Ընդհանուր ցուցումներ

- Ձեր նախագիծը պետք է գրվի `C`-ով:
- Ձեր նախագիծը պետք է գրվի `Norm`-ին համապատասխան: Եթե ունեք բոնուսային ֆայլեր/ֆունկցիաներ, դրանք ևս ստուգվելու են `Norm`-ով, և `Norm`-ի սխալի դեպքում կստանաք 0 միավոր:
- Բացառությամբ չսահմանված վարքագծերի՝ ձեր ֆունկցիաները չպետք է անսպասելի ավարտվեն (սեգմենտացիայի սխալ, bus սխալ, կրկնակի free և այլն): Եթե դա տեղի ունենա, նախագիծը կհամարվի ոչ ֆունկցիոնալ և գնահատման ժամանակ կստանաք 0 միավոր:
- Հատկացված դինամիկ բաշխվող հիշողությունն անհրաժեշտության դեպքում պետք է ամբողջությամբ պատշաճ կերպով ազատվի: Հիշողության արտահոսքերն անընդունելի են:
- Եթե նյութը պահանջում է, ապա պետք է հանձնել `Makefile`, որը ձեր ելակետային ֆայլը կկազմարկի պահանջվող ելքի՝ -Wall, -Wextra և –Werror դրոշակներով, իսկ ձեր `Makefile`-ը չպետք է վերակապի:
- Ձեր Makefile-ը պետք է պարունակի առնվազն `$(NAME)`, `all`, `clean`, `fclean` և `re` կանոնները:
- Ձեր նախագծի հետ բոնուսներ հանձնելու համար պետք է `Makefile`-ում ներառել `bonus` կանոն, որը կավելացնի բոլոր հնարավոր վերնագրերը, գրադարանները կամ ֆունկցիաները, որոնք արգելված են նախագծի հիմնական մասում: Բոնուսները պետք է լինեն առանձին ֆայլում՝ `_bonus.{c/h}`-ում: Պարտադիր և բոնուսային մասերի գնահատումը կատարվում է իրարից անկախ:
- Եթե նախագիծը թույլ է տալիս օգտագործել ձեր `libft`-ն, ապա պետք է պատճենեք դրա ելակետային ֆայլերը և համապատասխան `Makefile`-ը `libft` պանակում՝ իր համապատասխան `Makefile`-ով: Ձեր նախագծի `Makefile`-ը պետք է կազմարկի գրադարանը՝ օգտագործելով դրա `Makefile`-ը, իսկ հետո կազմարկի նախագիծը:
- Խրախուսելի է ձեր նախագծի համար թեստային ծրագրերի ստեղծումը, որոնք, սակայն, հաշվի չեն առնվի և չեն գնահատվի:

Դա հնարավորություն կընձեռնի հեշտությամբ թեստավորել ձեր և ձեր ընկերների աշխատանքը: Այս թեստերը հատկապես օգտակար կլինեն պաշտպանության ժամանակ: Ավելին, պաշտպանության ընթացքում ազատորեն կարող եք օգտագործել ինչպես ձեր, այնպես էլ այն ուսանողների թեստերը, ում աշխատանքը գնահատում եք:

- Ձեր աշխատանքը պետք է ներառել հանձնարարված `git` պահոցում: Կգնահատվի միայն `git` պահոցի աշխատանքը: Եթե նախատեսվում է, որ `Deepthought`-ն է գնահատելու ձեր աշխատանքը, ապա դա կարվի ձեր ընկերների գնահատումից հետո: Եթե `Deepthought`-ի գնահատման ընթացքում աշխատանքի ցանկացած մասում որևէ սխալ ի հայտ գա, գնահատումը կընդհատվի:

Գլուխ III

Պարտադիր մաս

Ֆունկցիայի անուն	get_next_line
Նախատիպ	char *get_next_line(int fd);
Հանձնվելիք ֆայլեր	get_next_line.c, get_next_line_utils.c, get_next_line.h
Պարամետրեր	#1. Ֆայլի նկարագրիչ, որից պետք է կարդալ
Վերադարձի արժեք	Կարդացած տողը, եթե ամեն բան նորմալ է, կամ NULL, եթե սխալ է տեղի ունեցել կամ էլ ոչինչ չկա կարդալու:
Արտաքին ֆունկցիաներ	read, malloc, free
Նկարագրություն	Գրել ֆունկցիա, որ վերադարձնում է ֆայլի նկարագրիչից կարդացած տող:

- `get_next_line` ֆունկցիան կրկնություններով, օրինակ ցիկլում կանչելը ձեզ թույլ կտա կարդալ ֆայլի նկարագրիչի վրա հասանելի տեքստը՝ տող առ տող, մինչև EOF-ին հասնելը:
- Ձեր ֆունկցիան պետք է վերադարձնի կարդացած տողը: Եթե այլևս ոչինչ չկա կարդալու կամ սխալ է տեղի ունեցել՝ պետք է վերադարձնի NULL:
- Համոզվեք, որ ձեր ֆունկցիան ճիշտ է աշխատում թե՛ ֆայլից, թե՛ ստանդարտ մուտքից կարդալիս:
- Հաշվի առեք, որ վերադարձված տողը ներառում է տողահատման `\n` նիշը, բացառությամբ այն դեպքի, երբ հասել եք ֆայլի վերջին, և տողը `\n` նիշով չի ավարտվում:
- `get_next_line.h` ֆայլը պետք է պարունակի գոնե `get_next_line()` ֆունկցիայի նախատիպը:
- Ավելացրեք բոլոր լրացուցիչ ֆունկցիաները `get_next_line_utils.c` ֆայլում:

Լավ սկիզբ կլինի, եթե իմանաք թե ինչ է ստատիկ փոփոխականը:

- Քանի որ ձեր `get_next_line()` ֆունկցիան պետք է ֆայլեր կարդա՝ ավելացրեք `-D BUFFER_SIZE=n` տարբերակը ձեր կազմարկիչի կանչին: Այդպիսով կորոշվի `read()`-ին փոխանցվող բուֆերի չափը:
  Այս արժեքը կձևափոխվի գնահատողների ու moulinette-ի կողմից:

Մենք պետք է կարողանանք կազմարկել այս նախագիծը -D BUFFER_SIZE դրոշակով ի լրացում սովորական դրոշակների, կամ առանց դրա: Կարող եք ինքներդ ընտրել կանխադրված արժեք:

- Նախագիծը պետք է կազմարկվի այսպես (բուֆերի չափին տրված է 42 արժեքը որպես օրինակ)՝ `cc -Wall -Wextra -Werror -D BUFFER_SIZE=42 <files>.c`
- Մենք հաշվի ենք առնում, որ `get_next_line()`-ն ունի չսահմանված վարքագիծ այն դեպքում, երբ ֆայլի նկարագրիչի ցույց տված ֆայլը փոփոխվել է նախորդ կանչից, մինչդեռ `read()`-ը դեռ չի հասել ֆայլի վերջին:
- Մենք նաև հաշվի ենք առնում, որ `get_next_line()`-ն ունի չսահմանված վարքագիծ երկուական ֆայլերի դեպքում: Այնուամենայնիվ, կարող եք տրամաբանական կերպով մշակել այս վարքագիծը, եթե ցանկանում եք:

Արդյոք ձեր ֆունկցիան աշխատու՞մ է, երբ BUFFER_SIZE-ի արժեքը 9999 է: Իսկ եթե BUFFER_SIZE-ի արժեքը մե՞կ է, իսկ 10000000-ի դեպքու՞մ: Ինչու՞:

Պետք է փորձել ինչքան հնարավոր է քիչ կարդալ get_next_line-ի ամեն կանչի ժամանակ: Եթե նոր տող հանդիպի, պետք է ընթացիկ տողը վերադարձնեք: Մի՛ կարդացեք ամբողջ ֆայլը ու հետո մշակեք յուրաքանչյուր տող:

Արգելված է.

- `libft`-ի օգտագործումը:
- `lseek()`-ի օգտագործումը:
- Գլոբալ փոփոխականների օգտագործումը:

Գլուխ IV

Բոնուսային մաս

Այս նախագիծը պարզ է: Բոնուսների համար շատ քիչ տեղ կա, բայց վստահ ենք, որ ձեր երևակայությունն անսահման է: Եթե պարտադիր մասը անթերի եք կատարել, ապա իհարկե կատարե՛ք բոնուսային:

Ահա բոնուսային մասի պահանջները.

- Իրականացրեք `get_next_line()`-ը միայն մեկ ստատիկ փոփոխականով:
- `get_next_line()`-ով պետք է կարողանալ մի քանի ֆայլի նկարագրիչներ կառավարել: Օրինակ, եթե ֆայլի 3, 4 ու 5 նկարագրիչները հասանելի են կարդալու համար, ապա `get_next_line()`-ը կարող եք կանչել մի անգամ 3-ի վրա, մի անգամ՝ 4-ի, ևս մեկ անգամ՝ 3-ի, հետո մի անգամ 5-ի վրա և այլն՝ առանց յուրաքանչյուր նկարագրիչի կարդալու շարանը (thread) կորցնելու:

Ավելացրեք `_bonus.[c\h]` վերջածանցը բոնուսային մասի ֆայլերին: Սա նշանակում է, որ, ի հավելում պարտադիր մասի ֆայլերին, պետք է հանձնել նաև հետևյալ երեք ֆայլերը՝

- `get_next_line_bonus.c`
- `get_next_line_bonus.h`
- `get_next_line_utils_bonus.c`

Բոնուսային մասը կգնահատվի միայն այն դեպքում, եթե պարտադիր մասը ԿԱՏԱՐՅԱԼ է: Կատարյալ նշանակում է, որ պարտադիր մասը ամբողջությամբ կատարված է և աշխատում է առանց սխալների: Եթե դուք չեք անցել ԲՈԼՈՐ պարտադիրը պահանջները, ձեր բոնուսային մասը ընդհանրապես չի գնահատվի:

Գլուխ V

Հանձնում և ընկերն ընկերոջը ստուգում

Հանձնեք ձեր առաջադրանքը `Git` պահոցում, ինչպես սովորաբար անում եք: Ստուգման ժամանակ գնահատվելու է միայն ձեր պահոցի պարունակությունը: Մի՛ վարանեք նորից ստուգել ձեր ֆայլերի անունները՝ համոզվելու համար, որ դրանք ճիշտ են:

Ձեր սեփական թեստերը գրելիս հիշեք, որ
1) Թե՛ բուֆֆերի չափը, և թե՛ տողի երկարությունը կարող են շատ տարբեր արժեքներ ընդունել:
2) Ֆայլի նկարագրիչը միայն սովորական ֆայլերի վրա ցույց չի տալիս:
Խելացի գտնվեք և փոխադարձաբար ստուգեք ձեր ընկերներին: Պատրաստեք բազմազան թեստեր պաշտպանության համար:

Առաջադրանքը հաջողությամբ ավարտելուց հետո մի՛ վարանեք `get_next_line()`-ը ձեր `libft`-ին ավելացնել: