Wstęp do programowania w C

Robert Ferens

Lista 6 - 24 listopada 2021

Zadanie 1

Napisz dwa proste programiki. Pierwszy (2 pkt) powinien w nieskończonej pętli czytać wartość time() (z time.h) i przy każdej zmianie wypisać "tik" ze znakiem końca linii. Program powinien być wyłączany przez ctrl+c.

Drugi (3 pkt) powinien jako parametr wywołania przyjąć 3 liczby z zakresów odpowiednio 0-23 0-59 0-59. A następnie wypisać podane liczby jako godzinę w formacie "gg:mm:ss". Później powinien w pętli czytać kolejne znaki z wejścia, i za każdym razem gdy pojawi się znak nowej linii(resztę znaków powinien ignorować), aktualizować godzinę dodając do niej 1 sekundę i wypisując nową godzinę(nie zapomnij znaku nowej linii po wypisaniu). Po jego uruchomieniu każde 'enter' powinno pokazywać czas zwiększony o 1 sekundę.

Na końcu (1pkt) połącz terminalem za pomocą pipe'a działania obu programów tak by wyjście pierwszego było przekazywane bezpośrednio na wejście drugiego za pomocą

```
LINUX:  ./prog1 | ./prog2 23 59 53
WINDOWS .\prog1.exe | .\prog2.exe 23 59 53
```

W wyniku tego wywołania powinien pojawić się zegar aktualizowany co sekundę do nowej godziny.

UWAGA! Normalnie prog1 nie przekazuje poprawinie wyjścia do wejscia prog2, wynika to z buforowania - gromadzenia danych w tablicy wewnętrznego bufora przed ich wypisaniem. Bufor wypisuje się domyślnie po znaku lub na koniec lini gdy działacie z terminala, natomiast dopiero po jego zapełnieniu lub zakończeniu programu, gdy piszecie do pliku lub inngo programu. Dlatego konieczne dla programu pierwszego będzie użycie fflush(stdout) po printf. Więcej o buforowaniu w "Buffer" oraz w opisie setvbuf() tutaj: `https://www.cplusplus.com/reference/cstdio/`

Dla uzyskania pozostałych 4 pkt wybierz coś z poniższej listy i doimplementuj do powyższych programów - napisz w komentarzu co wybrałeś:

- (2pkt) Gdy drugi program zostanie uruchomiony bez argumentów wczytaj aktualną godzinę za pomocą funkcji time() oraz localtime() z time.h . Punkt ten wymaga użycia struktur i wskaźników, więc lekko wykracza ponad materiał z tego tygodnia - może wymagać jakiejś chwili dociekania.
- (2pkt) W obecnej wersji pierwszy program zużywa dość sporo zasobów procesora(sprawdź w menadżerze zadań lub za pomocą top lub htop w linuxie). Spróbuj użyć funkcji (S)sleep,

w celu zredukowania zużycia zasobów - sprawdź czy zadziałało. Funkcja sleep w jest zależna od systemu i jest dostępna w Windows.h, albo w unistd.h. Przyjmuje też inne jednostki (ms,s) - doczytaj szczegóły o sleep w systemie w którym pracujesz.

- (4pkt) Spraw by zegar wyglądał ładnie - narysuj i zapisz w tablicach ascii-art dla każdej cyfry i dwukropka, a później odpowiednio z nich skorzystaj by wydrukować litery. Każdy druk możesz poprzedzić kilkunastoma znakami nowej linii tak by terminal wyglądał na wyczyszczony z samym tylko zegarem, ewentualnie użyj zależnych od systemów sposobu czyszczenia terminala. Mój przykład zegara:

```
 ###    I       000  000       ### ^^^^/
#   #  /I   @  0   00   0  @  #   #   /
  ##  / I      0   00   0       ##   /
 #      I   @  0   00   0  @  #   # /
#####  1#1      000  000       ### /
```

  Oczywiście liczę na ciekawe pomysły. Rozmiar liter dowolny, ale przy ustalaniu wielkości tablicy pamiętaj że każdy string zajmuje jeden więcej znak niż w zapisie (znak 0).

Zadanie 2

Dotychczas zawsze zakładaliśmy pewien maksymalny rozmiar tablicy już w momencie kompilacji(statyczny przydział pamięci). Program rezerwował całą tę pamięć podczas swojego działania ograniczając tym ilość dostępnej pamięci RAM dla innych programów.

Powiedzmy dla przykładu że chcielibyśmy napisać program, który potrafi dodawać macierze o maksymalnym rozmiarze 30 000x30 000, przeważnie będzie jednak działał na macierzach mniejszych niż 100x100. Nietrudno policzyć, że maksymalna macierz zajmuje $30000 \cdot 30000 \cdot sizeof(int) = 9 \cdot 4 \cdot 10^8 = 3.6GB$. Nie jest rozsądnym by małe obliczenia zabierały tak dużą pamięć roboczą gdy np. dla 100x100 wystarczy ok. 40kB. Potrzebny jest zatem mechanizm przydzilający pamięć podczas działania programu (dynamiczny przydział pamięci).

Machanizm ten w C odbywa się za pomocą funkcji malloc, calloc, realloc i free. Pierwsze dwie mają podobną funkcję - proszą system operacyjny o tyle pamięci ile od nich zażądamy i jeśli ją otrzymają zwrócą wskaźnik do niej. Ostatnia oddaje pamięć do systemu, kiedy jej już nie potrzebujemy - będziemy mogli jej użyć przy następnej alokacji. Należy zawsze pamiętać by całą pamięć jaką otrzymaliśmy od malloc lub calloc zwrócić przez free.

Realloc pozwala powiększyć miejsce na tablicę gdy okazuje się że jest go za mało. Dba on o także o to by wszystkie elementy tablicy pozostały na swoim miejscu względem początku. Nie jest jednak wykluczone że przeniesie je w inne miejsce w pamięci. Może też zmniejszyć ilość pamięci i w ten sposób zwolnić koniec tablicy którego już nie potrzebujemy.

Link do opisu technicznego: `https://pl.wikibooks.org/wiki/C/malloc`

Celem tego zadania jest zaznajomienie się z dynamicznym sposobem alokacji pamięci. Program ma operować na tablicy o dostosowującym się rozmiarze. Dla maksymalnego uproszczenia ćwiczenia program będzie wpisywał do tablicy całe wejście jakie otrzyma przez getchar aż do znaku EOF, a po odczytaniu EOF zaś je wypisze (jeżeli implementacja EOF stanowi problem można dla uproszczenia czytać aż do wybranego znaku np. '@'). W tym celu:

1. Stwórz zmienną max_size i zainicjuj ją na 10.
2. Stwórz zmienną $size$ i zainicjuj ją na 0.
3. Za pomocą malloc lub calloc pobierz wskaźnik char* tab na miejsce w pamieci na max_size elementów typu char.
4. UWAGA! wskaźnika tego możesz używać jak tablicy, ale uważaj by nie przekroczyć pamięci, czyli tab[5] pokaże na piątą wartość w zarezerwowanej pamięci oraz tab[5] jest tym samym co wskaźnikowe *(tab+5)
5. dodawaj do tablicy kolejne znaki z wejścia, przy każdym zwiększając $size$ o 1.
6. za każdym razem kiedy $size == max\_size$ podwój max_size i realloc'uj pamięć na nowy rozmiar tablicy. Po wykonaniu tej operacji wypisz komunikat o zwiększeniu ilości pamięci z podaną ilością nowej pamięci.
7. gdy napotkasz EOF wypisz zawartość tablicy, zwolnij pamięć i zakończ program.

Zadanie 3

Dostępne ze sprawdzaczką na skosie.