Wojciech Sitek

Koncepcja SOI – zadanie 6

# Treść zadania

* Napisać w języku C++ program, realizujący następujące funkcje systemu plików:
  + Tworzenie wirtualnego dysku,
  + Kopiowanie pliku z systemu Linux na dysk wirtualny,
  + Kopiowanie pliku z dysku wirtualnego na dysk systemu Linux,
  + Wyświetlanie katalogu dysku wirtualnego,
  + Usuwanie pliku z wirtualnego dysku,
  + Usuwanie wirtualnego dysku
  + Wyświetlenie zestawienia z aktualną mapą zajętości wirtualnego dysku z opisem: adres, typ obszaru, rozmiar, stan (np. dla bloków danych: wolny/zajęty).
* Napisać skrypt demonstrujący działanie programu

# Koncepcja rozwiązania zadania

1. Do obsługi stworzonego systemu plików będą wykorzystywane napisane krótkie skrypty w języku sh, wywołujące określone funkcje z programu w języku C++. Będą to komendy:
   * mkdisk [diskName(o)[[1]](#footnote-1)] – tworzenie wirtualnego dysku
   * put [filePath] [targetName(o)] [diskName(o)] – kopiowanie pliku z systemu Linux na dysk wirtualny
   * get [diskFileName] [targetPath] [diskName(o)] – kopiowanie pliku z dysku wirtualnego na dysk systemu Linux
   * ls – wyświetlanie dysków wirtualnych
   * ls [diskName] – wyświetlanie katalogu dysku wirtualnego
   * rmf [fileName] [diskName(o)] – usuwanie pliku z wirtualnego dysku
   * rm [diskName(o)] – usuwanie wirtualnego dysku
   * info [diskName(o)] – wyświetlanie zestawienia z mapą zajętości dysku
2. Skrypty będą opierały się na wywołaniu programu w C++ z odpowiednim argumentem (numerem funkcji w C++, implementującej odpowiednią akcję.
   * Nazwy funkcji będą identyczne jak skryptów.
3. Tworzenie wirtualnego dysku polega na stworzeniu ukrytego pliku (z prefiksem *.*)o nazwie podanej przez użytkownika bądź nazwie „*.x”*. Przy próbie stworzenia drugiego dysku wirtualnego o tej samej nazwie funkcja zwraca błąd. Wirtualny dysk ma następujące segmenty:
   * Informacje o dysku (rozmiar, liczba plików) i każdym z bloków pamięci ( adres, rozmiar, typ obszaru (plik/wolny blok), nazwa pliku (dla plików) )
     + rozmiar pliku będzie obliczany w funkcji tworzącej dysk i będzie on proporcjonalny do rozmiaru segmentu danych (im większy rozmiar miejsca na dane, tym większa potencjalna ilość bloków)
     + Dla każdego dysku zostanie wprowadzone ograniczenie ilości posiadanych plików ze względu na miejsce przeznaczone na informacje o dysku. Po osiągnięciu limitu nie będzie możliwe dodawanie plików na dysk.
   * Dane w kolejnych blokach
4. W przypadku braku pliku lub dysku o podanej nazwie lub istniejącym już pliku o podanej nazwie, program zwróci błąd i operacja kopiowania/wyświetlania/usuwania zostanie przerwana.
5. W przypadku braku pamięci na skopiowanie kolejnego pliku, program obliczy, czy brak miejsca na dysku jest spowodowany fragmentacją pamięci i gdy jest to konieczne, dokona próby defragmentacji pamięci poprzez wypełnienie danymi bloków pustych między plikami i zostawienie miejsca w ostatnim bloku pamięci.
6. Na dysku będzie wykorzystywany algorytm alokacji segmentów *worst fit*, ponieważ przy alokacji nie zależy nam na dużej szybkości algorytmu, tylko unikaniu fragmentacji, co ten algorytm zazwyczaj pozwala zapewnić. Bloki będą mogły znajdować się blisko siebie, ponieważ nie ma możliwości zmiany zawartości pliku, a jedynie ich przechowywanie, czyli kopiowanie oraz usuwanie.
7. Skrypt demonstracyjny będzie miał za zadanie przetestowanie funkcjonalności dysku wirtualnego i poprawności implementacji. Będzie on wykorzystywał wszystkie komendy shella, tworzył dyski wirtualne, a także tworzył pliki w systemie Linux i próbował kopiowania/usuwania/wyświetlania bloków/wyświetlania zawartości dysku/wykonywania niedozwolonych operacji, które powinny zakończyć się błędem.

1. *(o)* - optional [↑](#footnote-ref-1)