Sprawozdanie z Projektu nr 3

Przedmiot: Podstawy programowania [PRI]
Projekt nr 3

Adam Stec, grupa V
Nr albumu: 300261

1. Treść zadania
   Napisz program symulujący działanie bezprzewodowego łącza jednokierunkowego bezpośredniego pomiędzy dwoma użytkownikami. Symulacja ma polegać na przekazywaniu pakietów danych, z których każdy ma posiadać unikatowe oznaczenie, rozmiar i priorytet. Na podstawie dostarczonych danych powinna być tworzona kolejka priorytetowa (wykorzystaj drzewo BST). Dane mogą zostać przetransmitowane tylko w momencie otrzymania informacji (od użytkownika) o dostępności łącza. Należy założyć, że przesłanie danych następuje natychmiast po otrzymaniu informacji o dostępności łącza. Zarówno dane jak i informacje o dostępności łącza powinny być podawane z linii poleceń przez użytkownika pojedynczą instrukcją. Program powinien umożliwić wyświetlanie informacji o danych znajdujących się w kolejce do wysłania oraz danych dostarczonych do odbiorcy posortowanych względem rozmiaru.
   Program powinien działać z linii komend. Po uruchomieniu programu powinna pojawić się powłoka programu (shell), w której użytkownik będzie mógł podawać komendy. Dodatkowo po wpisaniu polecenia help powinna wyświetlić się pomoc programu wraz z opisem poszczególnych komend. Po wpisaniu help nazwa_polecenia powinna wyświetlić się informacja ze składnią danego polecenia. W przypadku gdy użytkownik pomyli składnię polecenia również powinna wyświetlić się pomoc dla danego polecenia.
   Testowanie powinno się odbyć na podstawie przygotowanych plików wsadowych z wykorzystaniem przekierowania strumienia do programu.

2. Opis działania oraz założenia
   Główne zadania programu to wczytywanie nazwy, rozmiaru i priorytetu plików ze standardowego wejścia i umieszczanie tych danych w kolejce opartej na drzewie BST. Program nie dopuszcza do podania dwóch plików o jednakowym priorytecie. Po wysłaniu, pliki zostają usunięte z drzewa i umieszczone w innej dynamicznej strukturze danych, jaką jest lista dwukierunkowa. Zgodnie z poleceniem, po wpisaniu „pomoc” użytkownik otrzymuje opis poszczególnych

funkcji programu.

3. Budowa i opis ważniejszych funkcji
   Program jest podzielony na 6 plików:
   - jeden główny o nazwie main.c;
   - drzewo.c zawierający wszystkie funkcje drzewa oraz listy dwukierunkowej;
   - powłoka.c zawierający ciała funkcji wywoływanych w main, korzystający z funkcji z pliku drzewo.c;
   - struktury.h deklaracje niezbędnych struktur;
   - fun_drzewa.h;
   - powloka.h.

   Najważniejsze oraz najbardziej skomplikowanie funkcje znajdują się w pliku drzewo.c. Znajdziemy tam implementację wszystkich niezbędnych funkcji takich jak dodawanie pliku do drzewa, usuwanie, oraz wysyłanie pliku z drzewa (usuwając go) do listy dwukierunkowej, umieszczając go we właściwe miejsce (zgodnie z porządkiem sortowania względem rozmiaru). Wszystkie nietrywialne funkcje są opatrzone komentarzami.
4. Kod źródłowy. Dwie przykładowe funkcje.

Funkcja dodająca plik do drzewa:

```
bool DodajPlik ( Plik *Pl, Drzewo *drz ){
    if ( CzyJuzJestPrio( Pl, drz ) == true ){
        fprintf( stderr, "Nie mozesz dodac pliku o zadanym
priorytecie, poniewaz inny plik juz ma taki priorytet
przesylania.\n");
        return false;
    }
    Wezel *nowy;
    nowy = NowyWezel( Pl );
    if ( nowy == NULL){
        fprintf( stderr, "Nie udalo sie utworzyc nowego
miejsca w kolejce przeslylania.\n");
        return false;
    }
    drz->rozmiar++;
    if ( drz->korzen == NULL )
        drz->korzen = nowy;
    else
        DodajWezel( nowy, drz->korzen );
    return true;
    //zwraca prawde, gdy pomyslnie dodano nowy plik
}
```

Dodawanie pliku do listy dwukierunkowej

```
void DodajPlikDoListy ( Plik *Pl, Lista *listaa ){
    Element *nowy;
    nowy = NowyElementListy( Pl );
    if ( nowy == NULL){
        fprintf( stderr, "Brak pamieci na urzadzeniu, do ktorego chcesz
przeslac plik.\n");
        exit( 1 );}
    Element *tymcz;
    Element *tymcz1;
    if ( listaa->glowa == NULL ){ //gdy lista jest pusta, sprawa jest
prosta
        listaa->glowa = nowy;
        listaa->rozm++;
    }
    else if ( nowy->F.rozmiar <= listaa->glowa->F.rozmiar ){ //dodawanie
elementu na poczatek listy
        listaa->glowa->poprz = nowy;
        tymcz = listaa->glowa;
        nowy->nast = tymcz;
        listaa->glowa = nowy;
        listaa->rozm++;
    }
    else if ( nowy->F.rozmiar > listaa->glowa->F.rozmiar ){
        while( (nowy->F.rozmiar > listaa->glowa->F.rozmiar) && listaa-
>glowa->nast != NULL ){// przesuwanie wskaznika
            listaa->glowa = listaa->glowa->nast;}
        if ( listaa->glowa->nast == NULL && (nowy->F.rozmiar > listaa-
>glowa->F.rozmiar) ){//dodawanie elementu na koniec listy
            listaa->glowa->nast = nowy;
            tymcz = listaa->glowa;
            nowy->poprz = tymcz;
            listaa->rozm++;
            while ( listaa->glowa->poprz != NULL ) //cofanie wskaznika
                listaa->glowa = listaa->glowa->poprz;
        }
        else { //dodawanie elementu w srodek listy
            tymcz = listaa->glowa->poprz;
            tymcz1 = listaa->glowa;
            tymcz->nast = nowy;
            tymcz1->poprz = nowy;
            nowy->poprz = tymcz;
            nowy->nast = tymcz1;
            listaa->rozm++;
            while ( listaa->glowa->poprz != NULL ) //cofanie wskaznika
                listaa->glowa = listaa->glowa->poprz;
        }
    }
}
```