Chart program

Készítette: Antal Roland

A program eredete:

Ez a program a Rendszerközeli Programozás egyetemi tárgyam projektfeladataként jött létre. A féléves projekt célja az volt, hogy jobban beletanuljunk a C programozási nyelvbe.

A program általános leírása:

A program 2 külön-külön, de egy időben futtatott verziója, azok egymás közötti kommunikációja lett megvalósítva mind file-on, mind pedig socketen kereszül.
A kommunikáció során egy 1bit színmélységű (emberi nyelven kétszínű) bmp kiterjesztésű képfájl legenerálásához szükséges adatokat állítja elő a program egyik példánya (küldő), majd ezeket az adatokat kapja meg (a két kommunikációs mód egyike által) a program másik példánya (fogadó), amely végül létrehozza a képfájlt, ezután pedig vár a következő feladatára. (A várakozásból kiléptethető a Ctrl + C gombkombináció lenyomásával.)
A kommunikáció csak úgy jöhet létre a küldő és fogadó között, ha mind a kettő azonos módban van futtatva (file – file vagy socket – socket)!
A generált bmp kép pedig egy véletlenszerűen változó mennyiség időbeli változását szemléltető grafikont ábrázol.

Rendszerkövetelmények:

A program a gcc 11.3.0 verzióját használja, így kérem ezt a verziót használja a stabil és megfelelő működés érdekében!
Emellett Linux rendszerre lett írva (Linux 22.04 OS), így csak azon működik!
A program fordítása a fentiek megléte mellett, a terminálból történik az alábbi parancs kiadásával:

```
gcc chart.c myfunctions.c -fopenmp -o chart
```

Működése ezek után a lehetséges kapcsolók megadásával állítható be.

Rendelkezésre álló kapcsolók:

- --version: A program jelenlegi verzióját, a legutóbbi módosítás dátumát és készítője nevét, valamint egyéb információkat tartalmazhat a programmal kapcsolatosan.

- --help: Információ, rövid leírás a program kapcsolóiról.
  (Hibás paraméterezéskor, vagy kapcsolók alkalmazása nélkül ez hívódik meg alapértelmezetten)

- -send/-receive: A program küldő/fogadó üzemmódban fog futni. Ha nincs megadva egyik sem akkor a -send az alapértelmezett.

- -file/-socket: A programok közötti kommunikáció módját lehet kiválasztani ezzel a kapcsolóval. Ha nincs megadva egyik sem, akkor a -file az alapértelmezett.

Megjegyzés: A kapcsolók sorrendje szabadon felcserélhető.

Hibakódok jelentései:

0. A program hiba nélkül végigfutott és leállt.
1. Parancssori argumentumokkal kapcsolatos probléma.
2. Nem chart a futtatható állomány neve.
3. Nem sikerült létrehozni a chart.bmp file-t.
4. Nem sikerült megnyitni a /proc könyvtárat.
5. Nem sikerült megnyitni a /proc könyvtárban talált proccesst.
6. Nem sikerült létrehozni a Measurment.txt file-t.
7. Nem sikerült megnyitni a Measurment.txt file-t.
8. Nem található fogadó folyamat a küldő számára file-on keresztüli kommunikációkor.
9. Nem sikerült új helyet foglalni a memóriában a Values tömb számára.
10. SIGUSR1 szignált kapott a program, miközben a küldési folyamat nem elérhető.
11. SIGALRM szignált kapott a program, tehát a szerver nem válaszolt időben.
12. Nem sikerült létrehozni a küldő oldali socketet.
13. Nem sikerült elküldeni a NumValues értékeket.
14. Nem válaszol a szerver oldali socket.
15. A szervertől visszakapott NumValues értéke nem megfelelő.
16. Nem sikerült elküldeni a Values tömb értékeit.
17. A szervertől visszakapott Values tömb mérete nem megfelelő.
18. Nem sikerült létrehozni a fogadó oldali socketet.
19. Nem sikerült a bindolás.
20. Nem érkezett meg a NumValues a szerver oldali sockethez.
21. Nem sikerült elküldeni a választ a küldő oldali socketnek a NumValues érkezése után.
22. Nem érkezett meg a Values tömb a szerver oldali sockethez.
23. Nem sikerült elküldeni a választ a küldő oldali socketnek a Values tömb érkezése után.
99. Ismeretlen jelet kapott a program és leállt..

Megjegyzés: A legutóbbi kilépési hibakódot a “echo $?” parancs kiadásával kérhetjük ki.

A program alprogramjai:

- void decToHex(int dec, unsigned char hex[4]);
  Ez az eljárás egy decimális értéket (int típusú szám: dec) alakít át hexadecimális számmá, majd ezt abban a 4 bájtos tömbben tárolja el, amelyet szintén paraméterként adtunk át neki (hex[4]).
  A bmp header részét megfelelő értékekre beállító changeHeader() eljárás segéd eljárása.

- void bitSwap(int first, unsigned char *header, unsigned char *array);
  A direktben megadott bmp header minta tömbben, a szükséges byte-ok cserélésére szolgáló segédeljárás, amelyet a changeHeader() eljárás használ.
  A megfelelő méretet, magasságot és szélességet definiáló byte-okat állítja be.
  Nem ez az eljárás számolja ki ezeket az értékeket, csak a csere elvégzéséért felel.
  A csere során minden esetben 4 egymást követő byte-ot kell lecserélnie, ehhez az első byte indexét kapja meg egész számként (first).
  Az array tárolja a 4 helyes byte-ot tartalmazó memóriaterület kezdőcímét, melyekre cserélni fogja a header karaktertömb meghatározott indexű byte-jait.

- int getSize(int side);
  Ez a függvény kap egy egész számot (side), amely a bmp kép magassága pixel darbszámban, majd ebből kiszámolja, hogy hány byte-on fog elférni a bmp_data teljes tartalma, végül hozzáad még 62-őt, amely a bmp_header mérete. Az általa visszaadott érték tehát a bmp képfile teljes mérete lesz.
  A függvény azzal is számol, hogy a bmp data minden sora 32 bittel oszható legyen. Szintén a changeHeader() eljárás használja a bmp_header tömb feltöltésére.

- void changeHeader(unsigned char bmp_header[62], int Decimal);
  Ez az eljárás fogja össze a fentebb található segédeljárásokat, felhasználva azokat a bmp_header karakter tömb helyes byte-okkal való feltöltésére.
  Az eljárás megkapja a módosítandó bmp_header[] karaktertömböt, valamint a Values adatsor méretét tartalmazó NumValues változót. Ezeket felhasználva a bmp_header[] minden byte-ja a megfelelő értékekre módosul.

- unsigned char hexSum(unsigned char hex, int dec);
  Ez a függvény egy hexadecimális és egy decimális szám összegét adja vissza hexadecimális számként, amelyre a bmp_data[] pixeltömb feltöltésekor van szükség.

- int myPow(int base, int pow);
  Egy hatványozás függvény amely visszaadja a hatvány értékét egész számként. A base az alap és pow a kitevő, azért tartalmazza ezt a program, hogy ne kelljen fordításkor a math könyvtárat is fordítani.

- void drawBMP(unsigned char bmp_data[], int data_size, int *Values, int NumValues);
  Ez az eljárás a changeHeader() eljáráshoz hasonlóan segédeljárásokat fog össze, amelyek segítségével először létrehoz egy megfelelő méretű karaktertömböt, ezt feltölti ‘0x00’, azaz hexadecimális 0 értékekkel, majd a szükséges byte-okat átállítja a kellő értékekre a bmp kép kirajzoltatásához.
  Bemenetként kapja a bmp_data[] karaktertömböt, melyet fel kell töltenie, a data_size egész számot, amely a bmp_data[] méretét tárolja, a Values pointert, amely az ábrázolandó értékeket tartalmazó memóriaterület kezdőértékére mutat és végezetül a NumValues egész számot, ami az ábrázolandó adatsor hosszát tárolja.

- void BMPcreator(int *Values, int NumValues);
  Ez az eljárás fogja össze a changeHeader() és drawBMP() eljárásokat, amelyek segítségével létrehozza a chart.bmp nevű képfile-t, majd beleírja a megfelelő értékeket. Ez a képfile csak a tulajdonos számára módosítható, a többi felhasználó számára csak olvasható lesz.
  Benetként megkapja a Values pointert, ami az ábrázolandó adatsor kezdőértékének memóriacímére mutat, valamint a NumValues-t, amely az adatsor hosszát tárolja egy egész számként.

- int Measurement(int **Values);
  Ez a függvény szolgál az adatsor létrehozásáért. Bemenetként kapja a Values pointer memóriacímére mutató pointert, majd az adatsor legenerálása után

ebben fogja eltárolni az adatsort, ezen kívül pedig visszaadja az adatsor méretét egész számként.
Az általa előállított adatsor véletlenszerű, hosszát a futtatás ideje (eltelt negyed óra aktuális másodpercei) befolyásolják.

- int FindPID();
  Ez a függvény azért felel, hogy megkeresse az éppen futó, chart nevű fogadó folyamat PID számát a /proc könyvtár alkönyvtárai között és visszaadja azt egész számként. Paraméter nélküli függvény.

- void SendViaFile(int *Values, int NumValues);
  A program alapértelmezett könyvtárában létrehoz egy Measurment.txt nevű szöveges file-t, amibe beleírja az átadott Values tömb értékeit “%d\n” formátumban. A feltöltés során felhasználja a NumValues vátozót amely a Values tömb hosszát tartalmazza egész számként.
  A txt feltöltése után, ellenőrzi hogy van-e futó fogadó folyamat, ha talál akkor kiküld egy felhasználóit 1-es (SIGUSR1) szignált annak.

- void ReceiveViaFile(int sig);
  Megnyitja a program alapértelmezett könyvtárában található Measurment.txt szöveges file-t és beolvassa a benne található egész számokat a Values tömbbe, melyet dinamikusan bővít minden sor után.
  Ezután meghívja a BMPcreator() eljárást, majd felszabadítja a lefoglalt memóriaterületet.

- void SignalHandler(int sig);
  Ez egy szignál kezelő eljárás amely 3 féle beérkező szignált képes kezelni.
  SIGINT esetén elköszön, SIGUSR1 esetén tájékoztatást ad arról, hogy a fájlon keresztüli küldés folyamat nem elérhető, SIGALRM esetén pedig tájékoztat arról, hogy a szerver nem válaszolt időben.

- void SendViaSocket(int *Values, int NumValues);
  Létrehoz egy kliens oldali socketet, majd a 3333-as porton és 127.0.0.1-es (localhost) IP címen keresztül elküldi a NumValues értéket a szerver oldali socketnek, ellenőrzi hogy a szerver válaszol-e, majd hogy a válaszából kapott érték mérete megegyezik e az adat méretével, majd ugyan ezt elvégzi a Values tömbbel is.
  Tehát a két paramétere a NumValues egész szám és a Values pointer.

- void ReceiveViaSocket();
  Létrehoz egy szerver oldali socketet, majd a 3333-as porton vár egy egész számot, amit utána visszaküld a kliens oldali socketnek ellenőrzésre, ezután egy egészeket tartalmazó tömböt vár és ugyan úgy visszaküldi ellenőrzésre miután megkapta.
  A kommunikáció befejeztével újabb üzenetre vár.