Rendszerközeli programozás projekt dokumentációja

Rendszerkövetelmények:

- 4GB RAM, AMD Fx™-8350 8 magos processzor, GTX 1650 (a fejlesztés ezen történt)
- Ubuntu 22.04.3 LTS 64 bites operációs rendszer (a fejlesztés ezen történt)
- GCC 11.4.0

A program működésének leírása:

A programot kétféleképpen lehet futtatni. Egy küldő és egy fogadó üzemmódban. A program szintén kétféle kommunikálhat, socketen és fájlon keresztül. A küldő fél állítja elő az adatokat, a fogadó fél pedig ábrázolja azokat. A fogadó fél egy chart.bmp fájlt generál le, ami egy grafikonon ábrázolja azokat a méréseket, amiket a küldőtől kapott.

A „gcc -o chart chart.c -fopenmp” paranccsal lehet lefordítani a programot.

A program lehetséges futtatási módjai és azoknak a magyarázatai:

./chart – Alapértelmezett üzemmód (send) és kommunikáció (fájl)

./chart --help – Kiírja az összes kapcsolót

./chart --version – A program verziószámát, elkészültének dátumát és a fejlesztő nevét írja ki

./chart -socket – Socketen keresztül történik a kommunikáció és küldő üzemmódban

./chart -file – Fájlon keresztül történik a kommunikáció és küldő üzemmódban

./chart -send – Fájlon keresztül történik a kommunikáció és küldő üzemmódban

./chart -receive – Fájlon keresztül történik a kommunikáció és fogadó üzemmódban

./chart -socket -send – Socketen keresztül történik a kommunikáció és küldő üzemmódban

./chart -socket -receive – Socketen keresztül történik a kommunikáció és fogadó üzemmódban

./chart -send -socket – Socketen keresztül történik a kommunikáció és küldő üzemmódban

./chart -receive -socket – Socketen keresztül történik a kommunikáció és fogadó üzemmódban

./chart -file -send – Fájlon keresztül történik a kommunikáció és küldő üzemmódban

./chart -file -receive – Fájlon keresztül történik a kommunikáció és fogadó üzemmódban

./chart -send -file– Fájlon keresztül történik a kommunikáció és küldő üzemmódban

./chart -receive -file – Fájlon keresztül történik a kommunikáció és fogadó üzemmódban

Ha ezek közül egyiket se használjuk, akkor hibaüzenettel tér vissza.

A program által visszaadott értékek magyarázata:

0 - Hiba nélkül tér vissza

1 - Hibásak az argumentum nevek

2 - Hibás a fájlnév (nem ./chart)

3 - Hiba a /proc mappa megnyitásában

4 - Hiba a /proc/status megnyitásában

5 - Hiba a statuscopy.txt megnyitásában

6 - Nincsen fogadó üzemmódban lévő process

7 - Nem sikerült elég memóriát foglalni a din. tömbhöz

8 - Nem található a /home/paleksz/Measurement.txt

9 - Socket létrehozási hiba

10 - Küldési hiba

11 - Fogadó hiba

12 - Nem egyezik a visszaküldött méret bájtban

13 - Nem egyezik a visszaküldött mérete a tömbnek

14 - A szerver leállt

15 - Hozzárendelési hiba

16 - SIGINT: Elköszönő üzenet

17 - SIGALRM: A szerver nem válaszol (időkereten belül)

Az elkészített alprogramok rövid leírása:

void help(): A metódus a lehetséges futtatási argumentumokat kiírja.

int Measurement(int Values):** Egy mutatóra mutató mutatót kap paraméterként, majd később ennek lefoglal egy adott területet, majd feltölti értékül, végezetül pedig visszaadja ezt a lefoglalt méretet egy int-ként. A méret függ az adott negyedórától eltelt másodpercek számától, de ha ez nem éri el a százat, akkor minimum ennyi lesz. A generált értékek pedig véletlenszerűen kisebbek, nagyobbak, vagy egyenlőek az előző elemhez képest.

int hatvany(int alap, int kitevo): Ez a függvény a megadott alap számot a megadott kitevő hatványára emeli, ahol a kitevő pozitív egész szám. A függvény minden ciklus iterációban megszorozza az eredményt az alappal, kezdetben az eredmény értéke 1.

void BMPformat(int x, char tomb[]): Ez a függvény az x egész számot egy 4 bájtos tomb karaktertömbbe konvertálja bitenként. A y változóval a bitműveletek során balról jobbra haladunk, az elem változó pedig segít nyomon követni, melyik bájtba helyezzük el az aktuális bit értékét. Az x szám bináris alakját bitenként másoljuk át a tomb tömbbe.

void BMPcreator (int x, char tomb[]): A BMPcreator függvény egy BMP fájlt hoz létre az Values nevű egész számokat tartalmazó tömb adatainak vizualizálására. A BMP fájl fejlécét és adatszekcióit állítja be, majd az eredményül kapott adatokat egy "chart.bmp" nevű fájlba írja.

int FindPID(): A FindPID() függvény egy Linux rendszeren próbál megkeresni egy adott nevű folyamat azonosítóját (PID), amelynek neve "chart". A folyamatok információit a /proc könyvtárban találja meg, ahol minden aktív folyamatnak saját könyvtára van a folyamat azonosítója (PID) alapján.

void SendViaFile(int *Values, int NumValues): Ez a függvény létrehoz egy "Measurement.txt" nevű fájlt, és beírja bele az Values tömb minden elemét. Ezután megpróbálja megtalálni a "chart" nevű folyamatot, és ha sikerül, akkor egy SIGUSR1 jelzést küld neki. Ha nem találja meg a folyamatot, hibát ír ki, és kilép a programból.

void memory_err(), **Dtomb *dt_create(Dtomb *self)**, **void *dt_add(Dtomb *self, int szam)**, **void *dt_destroy(Dtomb *self):** Ezek a függvények egy dinamikus tömb kezelésére szolgálnak. A dt_create létrehozza a tömböt és inicializálja annak kapacitását és hosszát. A dt_add hozzáad egy új elemet a tömbhöz, és automatikusan újraallokálja a memóriát, ha szükséges. A dt_destroy pedig felszabadítja a tömb által foglalt memóriát és a struktúrát is.

void ReceiveViaFile(int sig): Ez a függvény egy SIGUSR1 jelzést fogad, majd egy fájlt olvas be és a benne található adatokat egy dinamikus tömbbe menti. A beolvasott adatokat felhasználva kiszámolja a mérések különbségeit és létrehoz egy BMP fájlt ezekkel az adatokkal. Végül felszabadítja a dinamikus tömb által foglalt memóriát.

void SendViaSocket(int *Values, int NumValues): Ez a függvény egy socketen keresztül küld adatokat a szervernek. Először beállítja a szükséges változókat, majd inicializál egy socketet, és elküldi az adatokat a szervernek. A kapott válasz alapján további adatokat küld, majd lezárja a socketet.

void stop(int sig): Kiír egy üzenetet, hogy "A szerver leállt", majd leáll a program.

void ReceiveViaSocket():): Ez a függvény egy socketen keresztül vár adatokat a szervertől.