A Program Dokumentációja

A használandó fordítóprogram és annak szükséges kapcsolói

A program futtatási parancsa terminálból: *gcc main.c Functions.c -fopenmp -o chart*

Fordító programnak a gcc-t használjuk meg adjuk a futtatandó programok nevét kiterjesztéssel (fő program és hozzá szükséges alprogram a mi esetünkben) és a fordításhoz szükséges kapcsolókat, ezután entert ütünk és a program lefordul.

- *fopenmp* kapcsoló a verzió kezelőhöz miatt szükséges.

-*o chart* kapcsoló,h goy megfelelő néven tudjuk futtatni a programot.

Sikeres fordulás után tudjuk a programot futtatni terminálból ennek részleteiről a felhasználói útmutatóban olvashat részletesen.

Rendszerkövetelmények

- Operációs rendszer (OS): Ubuntu 64bit*
- Processzor (CPU): 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz 3.00 GHz (1 mag elég)
- Memória (RAM): 2048 MB RAM
- Videómemória (VRAM): 16 MB
- Tárhely (STO): 100 KB

* Ubuntu 64bit-en tökéletesen fut, ezért akinek ez fut a gépén nem kell aggódjon.

Felhasználói útmutató a programhoz

Ez egy C nyelvű program amely Linux alatt működik. 1gy 1bit színmélységű (két szín, ebben a verzióban fekete és fehér) bmp fájlt (1 pixel 1 bit) hoz létre, ami egy véletlenszerű idő változást mutató grafikon. Ezt file vagy socket kommunikációval hozható létre küldő és fogadó üzemmódban.

A programot *./chart* néven lehetet futtatni. Futtatásnál különböző módokon lehet futtatni kapcsolók segítségével.

--*version*: Kiírja a program verziószámát, elkészültének dátumát, időzónát licenszt copyright-ot és a fejlesztő nevét véletlenszerűen, mert több szálon fut. 0 a visszatérési értéke.

--*help*: futtatás lehetséges opcióiról és magyarázatáról ír ki információt. (Abban az esetben is megjelenik, ha rossz kapcsolót használunk) 0 a visszatérési értéke.

-*send*: Küldő üzemmód választása.

-*receive*: Fogadó üzemmód választása.

-*file*: Fájlon keresztüli kommunikáció választása.

-*socket*: Socket-en keresztüli kommunikáció választása.

Egy lehetséges futtatási lehetőség: *./chart -send -file*

Figyelem! Ha a programot kapcsolók nélkül futtatjuk, akor alapbeállítás fog elindulni, ami a -send – file, azaz küldő üzem mód file-on keresztüli kommunikáció.

A kapcsolók bármilyen sorrendben használhatók, nem megengedett kombinációja esetén is –help kapcsoló fog aktiválódni. (PL: -receive -send)

A program által visszaadott értékek magyarázata

A program visszatérési értékét a befejezést követően az *echo $?* parancs kiadásával kérdezhetjük le. A program a return 0/EXIT_SUCCES leállásán kívül mindegyikről értesíti a felhasználót és tartalmaz egy rövid leírást arról, hogy miért állt le (hiba üzenetként), ezek az exitek.

Visszatérési értékek magyarázata:

0 = A program sikeresen lefutott (return 0/EXIT_SUCCES) vagy a programot a felhasználó sikeresen leállította egy SIGINT(Ctrl+c) megszakító szignállal exit(EXIT_SUCCES). Az utóbbiról a felhasználó üzenetet is kap.

1 =A program SIGUSR1-es szignált kapott. "A fájlon keresztüli küldés szolgáltatás jelenleg nem elérhető!"

2 = A program SIGALRM szignált kapott. "A szerver (a megadott időkereten belül) nem válaszolt..."

3 = Nem megfelelő néven (nem chart) fordítottuk a programot, majd futtattuk.

4 = Fájl megnyitási hiba.

5 = Memória foglalási hiba.

6 = Jogosultság beállítási hiba.

7 = Nem található működő folyamat (process).

8 = Allokálási hiba.

9 = Socket létrehozási hiba

10 = Üzenet/adat küldési hiba

11 = Üzenet/adat fogadási hiba

12 = Kötési hiba.

13 = A munkakönyvtár elérési út nevének lekérése nem sikerült hiba.

Az elkészített alprogramok rövid leírása (cél, paraméterezés, visszatérési érték).

A Program különböző alprogramokat tartalmaz melyek a megfelelő üzemmód és a kommunikáció kiválasztást követően nem a –version és a --help kapcsoló, végrehajtódnak. A –version és – help leírása a „Felhasználói útmutató a programhoz” résznél olvasható.

Az alprogramok külön állományban vannak szervezve. Számos alprogram különböző értékekkel tér vissza melyek jelentéséről a „A program által visszaadott értékek magyarázata” részben olvashatott. Van, ahol nincs visszatérési érték ott a sikeres program lefutás után 0 a visszatérési érték vagy egy másik alprogram visszatérési értéke amely az alprogramon belül meg lett hívva.

- void SignalHandler(int sig);

  Ez egy szignálkezelő eljárás, ami háromféle szignált kezel. Ezek a szignálok a SIGINT (Ctrl+c), SIGUSR1(Terminálból adható ki: *kill -USR1* pid (a program pidje) paranccsal, de van olyan alprogram mely meghívja magának) és a SIGALRM.
  A program minden előtt felkeszül ezen szignálok fogadására. az int sig a signál elkapás miatt szükséges el. exit(EXIT_SUCCESS), exit(1) vagy exit(2) visszatérési értéket tartalmaz.

- int Measurement(int **values);

  Egy képzeletbeli szenzor által rendszeres időközönként mért értékekből egy függvénnyel állítjuk elő. Az értékek száma 100-900 közé tevődik. A kiindulási érték 0,

minden érték az előző értkhez mérten egy lehet, kisebb nagyobb vagy egyenlő a mérési szabályoknak megfelelően. int *Values értéket adunk át neki hívásnál. Hiba esetén exit(5) értékkel tér vissza, egyébként az előállított értékek számával tér vissza.
-send -file/-socket módban működik.

- void BMPcreator(int *Values, int NumValues);

A Measurement függvény által elállított értékekből egy chart.bmp képfájlt (1 pixel 1 bit) állít elő, ami 1 színmélységű (két szín, ebben az esetben fekete és fehér). Az első értek (0) mindig az elsőoszlop közepén helyezkedik el és ehhez igazodva a többi. Páros számú érték esetén a felős középsőre esik az első érték. Abban az esetben, ha az értékek túl lógnának a képen a legfeső vagy a legalsó sorban jelennek meg. A kép mérete adatokszámaXadatokszáma. Hiba esetén exit(13), exit(4) vagy exit(6) értékkel tér vissza, egyébként a program legenerálja a képet és fut tovább a program.
-receive -file/-socket módban működik. Automatikusan hívódik meg. Be van építve a többi alprogramba a hívása ott, ahol szükség van rá. Bementként az Values mutató egy egészeket tartalmazó tömb kezdőcímét kapja meg és a darabszámát.

- int FindPID();

A FindPID nevű processzus azonosítót kereső függvény. -1-es értékkel tér vissza a ha a program nem talált más .chart nevű folyamatot a saját magán kívül, ha talál akkor annak a PID értékével tér vissza. A saját Pid-jét figyelmen kívül hagyja. Linux fájlrendszer gyökerében lévő ”/proc” könyvtárnak az alkönyvtáraiban található ”status” között keres chart nevű futó prosszeket. Nincs szüksége bementre.
Automatikusan hívódik meg. Be van építve a többi alprogramba a hívása ott, ahol szükség van rá.
A *ps -aux | grep chart* paranccsal kérhetjük le saját PID-ünket a megfelelő sor kiolvasásával.

- void SendViaFile(int *Values, int NumValues);

Bementként az értékeket kapja meg és a darabszámát. Hiba esetén exit(4), egyébként exit(7) vagy SIGUSR1 exit értékével tér vissza. A Values mutató egy egészeket tartalmazó tömb kezdőcímét kapj meg és a darabszámát. Az eljárás létre hoz egy ”Measurement.txt” nevű az adott felhasználó saját alapértelmezett könyvtárában és soronként 1étrék és úgy írja bele a tömbben lévő értékeket a fájlba.
-send -file módban működik.

- void ReceiveViaFile(int sig);

Paramétre szükséges, de nem használjuk úgy hívjuk, meg hogy felkészült legyen szignálra. Hiba esetén exit(4), egyébként nincs visszatérési értéke. (Programon belül meghívott programnak azonban lehet így lekérdezésnél azt láthatjuk.) Az eljárás beolvassa a SendViaFile eljárás által generált ”Measurement.txt”-t és átadja a dinamikus memória foglalással eltárolja, majd meghívja a BMPcreator eljárást azután felszabadítja a memóriaterületeket.
-receive -file módban működik.

- int SendViaSocket(int *Values, int NumValues);

Ő a kliens. Bementként az értékeket kapja meg és a darabszámát. Hiba esetén exit(9), exit(10), exit(11) vagy SIGALRM szignál exit-jével tér vissza, egyébként 0.
UDP protokoll segítségével a localhost (IPv4 cím: 127.0.0.1) 3333-as portját figyelő fogadó üzemmódú ReceiveViaSocket függvénnyel kommunikál. Először elküldi a NumValues változó értékét (32 bites fix pontos egészként), majd várja a választ, ha a válasz mérete megegyezik az elküldöttel akkor elküldi a Values címen kezdődő tömb NumValues darab int típusú értékét küldi át egyetlen üzenetben a fogadónak, és ha válasz megegyezik a küldött mérettel akkor a program 0 értékkel leáll, ha nem akkor pedig a megfelelő exit-tel. Ha nem kap választ 1 másodpercen belül a program SIGALARM-ot küld és annak az értékével áll le.
-send -file módban műdökik.

- int ReceiveViaSocket();

Ő a szerver. Nincs bemente. Hiba esetén exit(9), exit(10), exit(11) vagy exit(12) tér vissza, egyébként 0.
Egy végtelen ciklusban UDP szegmenseket vár a 3333-as porton. Az első kapott szegmensben mindig egy 4 bájtos (int változó értéke, ami a darabszám lesz). Az eljárás nyugtaként visszaküldi a kapott értéket a küldő üzemmódú kliensnek, ezután dinamikusan lefoglalja a memóriát annyi egész számnak (int) amennyi a beérkezett szám értéke. Ott fogja eltárolni a második üzenetben kapott adatokat, aminek a bájtban megadott méretét nyugtaként visszaküldi a küldőnek. A kapott adatokkal meghívja a BMPcreator eljárást. A kép létrehozása után felszabadítja a lefoglalt memóriát és egy újabb üzenetre vár.
-receive -socket módban működik.