Rendszerközeli programozás
Projekt feladat

Készítette: Lengyel Szilárd Ferenc ISG82L

1. A használandó fordítóprogram és annak szükséges kapcsolói.

2. Rendszerkövetelményeket.

3. Felhasználói útmutatást a programhoz.

4. A program által visszaadott értékek magyarázatát.

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

1. A használandó fordítóprogram és annak szükséges kapcsolói:

-A program teljeskörű lefordításához GCC (GNU Compiler Collection) 11.3.0-ás verziója szükséges.

-A program lefordításához használatos „Makefile” a mappán belül megtalálható.
A használata: bármilyen shellben a „make” parancs beírásával.

2. Rendszerkövetelményeket:

-A program a legújabb (ver. 22.04 LTS) Ubuntun lett megírva és tesztelve, így a használatóhoz is ez ajánlott.

-Megközelítőleg <2MB tárhelyre van szüksége.

3. Felhasználói útmutatást a programhoz:

Használata: **./chart [OPTIONS]**

Opcionális paraméterek [OPTIONS]:

--version: A program verzió számáról ad részletesebb információt.

--help: Előhívja ezt a segítő üzenetet.

-send: Küldő üzemmódba vált. **(alapértelmezett)**

-receive: Fogadó üzemmódba vált.

-file: Fájlon keresztüli kommunikációra vált. **(alapértelmezett)**

-socket: Socketen keresztüli kommunikációra vált.

A paraméterek rendeltetés szerinti használatához illő megjegyezni egy–két dolgot:

-Nem lehet használni egyszerre több ismétlődő kapcsolót:
./chart -send -send
./chart -file -file
etc.
-Nem lehet használni egyszerre 2 ellentétes kapcsolót:
(file-socket)
(receive-send)
./chart -socket -file
./chart -receive -send
-Ha nem a makefile-al fordítjuk le a programot, hanem kézzel a terminálon keresztül akkor fontos tudni azt, hogy csakis ./chart néven lehet futtatni a programot, valamint 3 különböző kapcsolóra is van szükség (A Projekt.c programot kell lefordítani).

```
gcc -o chart Functions.c Projekt.c -lm -fopenmp
```

4. A program által visszaadott értékek magyarázatát:

0 – A program hiba nélkül futott le.
1 – A program neve nem chart
2 – Hibás paraméter kezelés
3 – Nincs fogadó üzemmódú folyamat
4 – Socket létrehozás sikertelen
5 – Sikertelen küldés
6 – Sikertelen fogadás
7 – A fogadott valamint küldött érték különbözik
8 – Kapcsolódási hiba
9 – Fájlon keresztüli küldés szolgáltatás nem elérhető
10 – Szerver nem válaszol
170 – Version (easter egg)

5. Az elkészített alprogramok rövid leírását (cél, paraméterezés, visszatérési érték):

```
void Mod_handling(int argc, char **argv, int **modes)
```

-A függvény a parancssori argumentumokat várja, valamint a darabszámát. Ezek mellett egy tömböt is vár, ahova a pszeudócím szerinti paraméter átadással helyben módosítja a tömb méretét és értékeit.

-A célja, hogy a megadott paraméterekben kiszűrje a hibalehetőségeket, és ezekre készít egy „kapcsoló” tömböt.
-A hibák esetén leáll és ezeknek megfelelő hibakódot ad vissza, de ha helyesen fut le, akkor a módosított tömbbel tudunk tovább dolgozni.

```
int idokezeles()
```

-Ez a függvény az aktuális időhöz mértan állít elő egy 4 bájtos intet, amely az előállított darabszámokat fogja reprezentálni.
-A „mért” értékek darabszáma megegyezik a hívás pillanatában az adott negyed órából eltelt másodpercek számának és a 100-nak a maximumával!
-Ez a függvény nem vár semmilyen paramétert.

```
int Measurement(int **Values)
```

-Ez a függvény egy tömböt vár paraméterként, majd ezt a tömböt pszeudócím szerinti paraméterátadással fogja helyben módosítani, reallocálni, valamint az értékeit megváltoztatni.
-Ez a függvény felhasználja az időkezelés függvényt, így kapjuk meg az értékeket.
-Egy 3 állapotú 1 dimenziós bolyongást implementál, azaz véletlenszerűen előállítja egész számok egy véges sorozatát, ahol bármely két szomszédos elem különbségének az abszolút értéke maximum 1

```
void BMPcreator(int *Values, int NumValues)
```

-Az első paraméter az ábrázolandó értékeket tartalmazó tömb kezdőcíme, a második pedig az előbbi tömbben tárolt egész értékek száma. A létrehozandó képfile neve chart.bmp, abban a könyvtárban jön létre, ahol a fogadó program el lett indítva.
-Ezt az 1 dimenziós 3 állapotú bolyongást ez a függvény implementálja a képbe.
-Ezek az értékek bájtonként íródnak bele (narancssárga színnel)

```
int FindPID()
```

-Ez a függvény nem vár semmilyen paramétert.

-Megkeresi a ’chart’ nevű folyamat PID-jét és ha ez nem egyezik meg a saját PID-jével, akkor ezt az értéket visszaadja, különben -1-et ad vissza.

```
void SendViaFile(int *Values, int NumValues)
```

-Ezt az eljárást a Measurement függvénnyel együtt hívjuk meg, így az általa módosított tömbbel és az elemszámmal dolgozik tovább.
-Ez az eljárás a felhasználó „HOME” könyvtárába létrehoz egy Measurements.txt-t és abba beleiratja a tömbben lévő értékeket.
-Majd megnézi, hogy van-e fogadó és fileon keresztüli kommunikációban futó folyamat. Ha van akkor küld arra a PID-re egy signált, ha nincs akkor pedig leáll az ennek megfelelő hibakóddal és hibaüzenettel.

```
void ReceiveViaFile(int sig)
```

-Az eljárás paraméterét nem fogjuk felhasználni.
-Ha megkapja a SendViaFile nevű függvénytöl a SIGUSR1-es signált, akkor a felhasználó „HOME” könyvtárából beolvassa a Measurements.txt-t és eltárolja ezeket az értékeket egy dinamikusan lefoglalt memória területre, majd meghívja a BMPcreator eljárást.
-Nem tér vissza semmilyen értékkel.

```
void SendViaSocket(int *Values, int NumValues)
```

-Ezt az eljárást a Measurements függvénnyel együtt hívjuk meg, így az általa módosított tömbbel és az elemszámmal dolgozik tovább.
-Ez az eljárás egy UDP klienst valósít meg, amely két küldéssel juttatja át a megfelelő adatokat a fogadó UDP szervernek.
-Első küldésre az előállított értékeket küldi el, majd második küldésre, az értékeket tároló tömb kezdőcímét.
-Az eljárás a küldések, valamint a fogadások mellett folyamatosan ellenőrzi a kezelt csomagok helyességét.
-Ha bármi hiba adódik, a program leáll az annak megfelelő hibakóddal és hibaüzenettel.

```
void ReceiveViaSocket()
```

-Ez az eljárás valósítja meg az UDP szervert, amely elindítás után folyamatosan várja a 3333-as porton a localhost-tól a csomagokat.
-Először az előállított értékek számát várja, majd utána az eltárolandó tömb első elemének a memóriacímét.
-Ezekre az értékekre, meghívja a BMPcreator eljárást, majd várja a következő kérést.

```
void SignalHandler(int sig)
```

-Az eljárás paraméterként kap egy számot, majd azt beazonosítja, hogy melyik signal értékével egyezik meg.
-A főprogramot, valamint néhány eljárást felkészíti a SIGUSR1-es, SIGALRM valamint SIGINT jelekre.
-A SIGUSR1-es jelre a program leáll helyesen, a többi jelre, pedig egy hibaüzenettel, valamint hibakóddal leáll.