Rendszerközeli programozás - dokumentáció a projektfeladathoz.

# 1. Program fordítása

A program futtatásához szükséges egy Linux operációs rendszerre és a GCC fordítóra. Ha az előzőek adotttak akkor fordítsuk le a programot a

*gcc projekt.c -fopenmp*

paranccsal. Ha nem egyben fordítjuk az **-fopenmp** fájllal akkor a program nem lesz képes több szálon futni.

# 2. A program futtatása

A fordítás után a programunkat a

*./a.out*

paranccsal tudjuk futtatni és ha nem adunk meg kapcsolókat akkor kilistázza nekünk az aktuális könyvtár tartalmát (lilával a mappákat, kékkel a reguláris fájlokat). A könyvtár nevének megadásával tudjuk megnyitni a kiválasztott könyvtárat.

A .. könyvtárnév megadásával tudunk visszalépni az előző könyvtárba.

A program mindig kilistázza az aktuális könyvtárunk tartalmát egészen addig ameddig nem reguláris fájlt adunk meg bemenetként. Ha egy nemlétező fájl/mappa nevét adjuk meg akkor a program visszatér egy hibaüzenettel és vár addig ameddig új bemenetet meg nem adunk. Amint létező képfájl nevet adunk meg a program tovább lép és azzal kezd el dolgozni.

A programot tudjuk parancssori argumentummal is futtatni, amelyiket a *–help* kapcsolóval tudunk megjeleníteni.

*./a.out [argumentum1] [argumentum2] ...*

Ha hibás argumentumot (vagy argumentumokat) adunk meg akkor a program hibát jelez.

Ha több argumentumot adunk meg akkor mindig az elsővel fog dolgozni a program.

Amennyiben valós bmp fájl nevet adtunk meg, a programunk beolvassa annak bájtjait, letárolja majd dekódolja azokat.

Dekódolás után a program a dekódolt szöveget egy webszerverre továbbítja és ha sikeresen megtörtént a bejegyzés a program visszatér egy üzenettel majd leáll.

# 3. A program által visszaadott értékek

- *Hiba! Adj meg létező képfájlnevet vagy kapcsolót(--help)*

A program arra figyelmeztet hogy a megadott argumentummal nem tud dolgozni.

- Hiba! Nem létező állomány!

A fájl  tallózásakor hibás file nevet kapott bemenetként ezért újra bekér egy nevet.

- Hiba a memóriaterület lefoglalása közben!

Nem sikerült a memória allokáció.

- Hiba történt a socket létrehozása közben!
- Kapcsolat létrehozása közben hiba történt!
- Az adat küldése során hiba történt.
- Az válasz fogadása során hiba történt.

A Post metódus során kapott hibákkal kapcsolatos hibaüzenetek.

- A program túl sokáig futott!

Ha filekezelés és a dekódolás több idő mint 1 másodperc a program leáll.

- A dekódolást a CTRL+C kombinációval nem tudod megszakítani!

Egy figyelmeztetés amit akkor kapunk ha le akarjuk állítani a programunkat az fut.

- *A szöveg sikeresen elküldve!*

A program lefutott és sikeresen továbbította a dekódolt üzenetet.

# 4. Alprogramok

*BrowseForOpen()*

Az egész függvény egy while ciklusból áll ami addig fut ameddig nem kap egy képfile nevet. Minden megadott input után kilistázza az aktuális mappa tartalmát(a rejtett objektumokat is). Visszatérési értéke egy bináris file leírója.

*WhatToDo()*

Egy szignálkezelő metódus ami képes kezelni a SIGINT és SIGALRM szignált.

*ReadPixels(int f, int* NumCh)*

Ebben a metódusban tároljuk le a bájtokat a képből. Két paramétere van, az egyik (f) a bináris fájl leírója, a másik(NumCh) a kódolt karakterek számát tartalmazó pointer, amit az alprogramunk módosítani fog.

Először kiolvassuk a kódolt karakterek számát és a kép méretét. (a kép első 54 bájtja tartalmazza ezt) .

Ezután beolvassuk egy dinamikus tömbbe a bájtokat, majd megfordítjuk a pixelek(3 bájt) sorrendjét mivel azok little-endian sorrendben voltak.

A függvény visszatér a bájtokat tároló tömb címével.

*Unwrap(char* Pbuff, int NumCh)*

A Pbuff a letárolt bájtokat tartalmazó tömb címe.

A szabály alapján dekódolja a program a pixeleket bitműveletek segítségével egy for ciklusban.

Külön kezeli azokat a karaktereket amelyek nem találhatóak meg az ASCII táblában és lecseréli őket az ASCII megfelelőjükre.

A ciklus végén egy utolsó szűrő található, hogy csak az értelmes karakterek tárolja el, majd visszatér a tömb címével

*Post(char *neptunID, char *message, int NumCh)*

Ez a függvény a dekódolt üzenet átküldésére szolgál.

A neptunID tartalmazza a készítő neptunkódját, a message a dekódolt üzenetet és a NumCh továbbra is a kódolt karakterek számát.

Létrehozzuk a socketet és a kapcsolatot a webszerverrel, majd egy buffer változóba belerakjuk az átküldendő szöveget.

Átküldjük az üzenetet és ha nem érkezik hibaüzenet akkor a konzolban megjeleni a sikeres átküldést jelentő üzenet.