Projekt dokumentáció

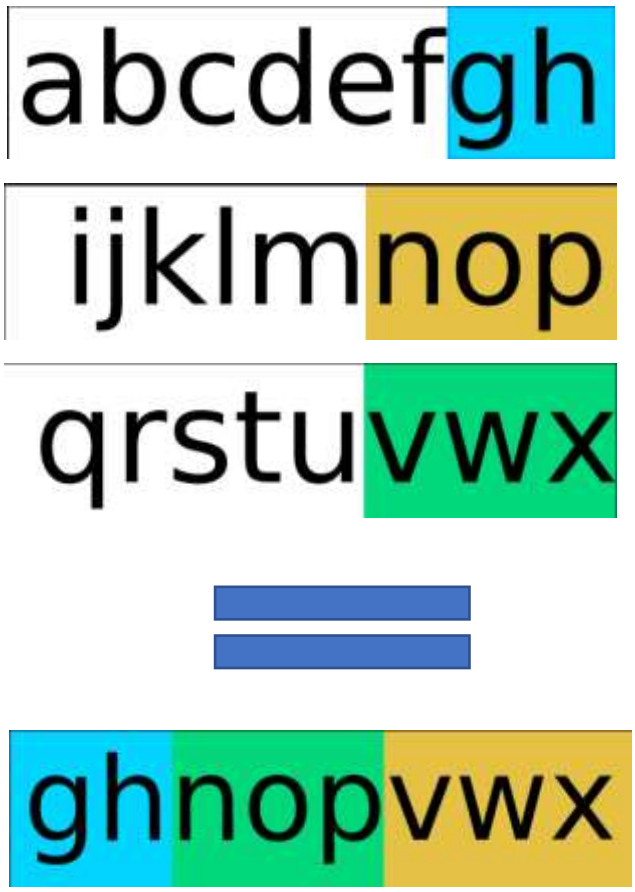
Rendszerközeli Programozás – Tarr Márton

Leírás

Linux környezetben futtatható szoftver, amely egy TrueColor BMP-fájlba ágyazott titkos szöveget dekódol, majd a szöveget egy web-szerverre továbbítja.

A szöveg kódolása a következő:

A program tömörítetlen, 24bites bitmap képfájlba rejtett szöveges információ kibontását fogja végzi. A szöveg elrejtése a következőképpen történt. Egy valódi képfájl minden pixelének 3 RGB színkomponense együtt egy ASCII karaktert (vagy egy UTF-8 bájtot) kódol úgy, hogy színkomponensek csak nagyon kis mértékben térjenek el az eredetitől (amit az emberi szem nem is igazán érzékel). A bináris pixel adat első bájtjának utolsó (legkisebb helyi értékű) két bitje az ASCII kód két legelső (legnagyobb helyi értékű) bitjét jelenti. A pixel második bájtjának utolsó 3 bitje a karakterkód következő 3 bitjét rejti. Végül a pixel harmadik színkomponensének utolsó 3 bitje a kódolt ASCII karakter utolsó három bitjét adja meg. Tehát egy általános pixel „abcdefgh ijklmnop qrstuvwx” bitjeiből kell előállítani a „ghnopvwx” bitek alkotta ASCII kódot.



Fordítóprogram, és annak kapcsolói

Választott fordítóprogram : GCC

A Program indításához szükséges lépések:

1. Ellenőrizze, hogy megtalálható minden szükséges fájl
    a. projekt.c
    b. mydef.h
    c. colors.h
    d. cpu.bmp
2. A sikeres működéshez szükség van a header fájlok a fő-programegységgel történő fordítása az alábbi módon történik:

```
gcc projekt.c mydef.h colors.h -o projekt -fopenmp
```

- **gcc** – C-fordító
- **projekt.c** – a main függvényt tartalmazó fájl neve
- **mydef.h colors.h** – header állományok
- **-o projekt** – a kapcsoló hatására .c kiterjesztésű forrás állományból .o kiterjesztésű tárgykódú állomány készül, „projekt” elnevezéssel
- **- fopenmp** – az OpenMP könyvtár használatához szükséges kapcsoló

3. Az előző pontban felhasznált parancs következményeként elkészült a futtatható állomány, amely „projekt” elnevezéssel található meg ugyanabban a könyvtárban ahol a parancsot kiadtuk. A futtatáshoz a **./projekt** parancsot kell használnunk

Használati útmutató

A program több parancssori argumentum kezelésére is képes.
Ezeknek megadása az alábbi módon történik:

```
./projekt [arg]
```

Ahol az [arg] az alábbi parancsok egyike:

1. Futtatás parancssori argumentum nélkül
    - Meghívásra kerül a BrowseForOpen függvény, amely egy parancssori könyvtár bejárási lehetőséget biztosít a dekódolni kívánt .bmp kiterjesztésű fájl megkeresésére az alábbi módon:
        - Minden parancs végrehajtása az „Enter” billentyű segítségével történik
        - Amennyiben a felhasználó egy, a felsorolt könyvtárak nevei közül választ, és ezt begépeli, a program belép az adott könyvtárba
        - A könyvtárak közötti visszalépés a „back” paranccsal történik
        - Amennyiben a begépelt állomány neve egy fájl, azt a program megpróbálja megnyitni
2. -- version
    - A parancssoron kiírásra kerül a program verzió száma, a fejlesztő neve, valamint az elkészülés dátuma. Ezek után a végrehajtás befejeződik, a program kilép.
3. – help
    - A parancssoron kiírásra kerülnek a futtatás lehetőségei
    - Lásd 1., 4., 2.
4. .bmp fájl neve
    - pl.: **./projekt cpu.bmp**
    - A program megkísérli beolvasni a parancssori argumentumként megadott fájlt, azt dekódolni, és a szöveget a webszerverre feltölteni

A program által visszaadott értékek

- **1** „File could not be opened”
  Ok: A megadott fájl megnyitása során hiba történt, nem nyitható meg
  Következmény: A program visszatér 1-es hibakóddal, majd kilép
- **2** „Version or help information”
  Ok: A felhasználó a –version vagy –help parancssori argumentumokat használta
  Következmény: A program kiírja a kért információt, visszatér 2-es kóddal, majd kilép
- **3** „No hidden text”
  Ok: A megadott képfájlban nem található rejtett szöveg
  Következmény: A program visszatér 3-as hibakóddal, majd kilép
- **1** „Memory could not be allocated”
  Ok: Sikertelen memóriafoglalás
  Következmény: A program visszatér 4-es hibakóddal, majd kilép
- **5** „Execution timed out”
  Ok: Időtúllépés, a szignál kezelő függvény SIGALRM szignált kapott paraméterként
  Következmény: A program visszatér 5-ös hibakóddal, majd kilép
- **A Post függvényhez tartozó visszatérési értékek:**
  0 **„Post succesfull! Text sent.”**
  Ok: A szöveg sikeresen feltöltésre került a webszerverre
  Következmény: A program visszatér 0-ás hibakóddal, majd kilép
  6 **„Text could not be posted!**”
  Ok: A szöveget nem sikerült a webszerverre feltölteni
  Következmény: A program visszatér 6-os hibakóddal, majd kilép
  7 **„Error opening socket”**
  Ok: A socket megnyitása sikertelen volt
  Következmény: A program 7-es értékkel kilép
  8 **„Error connecting”**
  Ok: A csatlakozás sikertelen volt
  Következmény: A program visszatér 8-es hibakóddal, majd kilép
  9 **„Wrong file format”**
  Ok: A program nem .bmp kiterjesztésű fájlt kapott
  Következmény: A program f visszatér 9-es hibakóddal, majd kilép

A programban található függvények

Az alábbi függvények mindegyike a **mydef.h** állományban található

Unwrap

```
char* Unwrap( char *Pbuff,
              int  NumCh )
```

Dekódolja a paraméterként kapott bmp fájl pixel array részét tartalmazó memóriacímét, valamint az üzenet hosszát. A tömb feldolgozása a rendelkezésre álló összes processzormagon párhuzamosan, közel azonos terheléssel történik.

Paraméterek

Pbuff a bmp fájl pixel array részét tartalmazó memóriacím

NumCh a kódolt üzenet hossza

Visszatérési érték

A dekódolt szöveg címe

ReadPixels

```
char* ReadPixels( int  fd,
                  int* NumCh )
```

Beolvassa egy tömörítetlen TrueColor bmp fájl tartalmát, majd a pixel array részét elhelyezi egy megfelelő méretű dinamikusan méretű tömbben. Amennyiben a folyamat 1 mp-nél tovább tartott, a program le fog állni.

Paraméterek

fd file descriptor

NumCH rejtett szöveg hossza

Visszatérési érték

A bmp file pixel array tartalmának címe

Lásd Továbbá

Unwrap

BrowseForOpen

```
int BrowseForOpen()
```

Karakteres felületű tallózás. Használata: Kiírja a parancssori felületre a felhasználó alapértelmezett könyvtárának tartalmát (a rejtett könyvtári objektumokat is). - A választott könyvtár nevének begépelése, majd az Enter billentyű megnyomása megnyitja a könyvtárat - a "back" parancs egyel vissza lép a könyvtár hierarchiában - amennyiben a bemenet egy reguláris fájl, azt megpróbálja megnyitni

Visszatérési érték

A bináris olvasásra megnyitott fájl descriptor

Post

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

A dekódolt szöveget egy webszerverre továbbítja

Paraméterek

neptunID a hallgató neptun kódja

`char*`

message a dekódolt üzenet

`char*`

NumCh az üzenet hossza

`int`

Visszatérési érték

- **0** amennyiben a webszerver fogadta az üzenetet
- **6** amennyiben nem fogadta, vagy hiba lépett fel

WhatToDo

```
void WhatToDo( int sign )
```

SIGALARM és SIGINT szignálok kezelése

Paraméterek

sign a szignál száma

`int`