Rendszerközeli programozás

Projekt feladat

Hozz létre Linux alatt egy olyan C nyelvű programot, amely egy TrueColor bmp fájlba beágyazott „láthatatlan” szöveget kicsomagol és azt egy HTTP POST metódussal egy megadott web szerverre eljuttatja! A program fejlesztése során az alábbi funkciókat implementáld a leírásnak megfelelően:

1. feladat

Hozz létre a C program fő programegységét, amely a parancssori argumentumokat is kezeli! Ha a „`--version`” parancssori argumentummal indítjuk el a programot, akkor írja ki a program verziószámát, elkészültének dátumát és a fejlesztő nevét! Ha a „`--help`” argumentummal indul a program, akkor adjon tájékoztatást a felhasználónak a futtatás lehetséges opcióiról! A fenti esetekben a végrehajtás fejeződjön is be! Ha az előzőektől eltérő parancssori argumentum van megadva, akkor a program ezek közül az elsőt tekintse egy fájl nevének, amivel később dolgozni fog! Ha nincs megadva parancssori argumentum, akkor a program a bemeneti fájlt később fogja meghatározni. A főprogram fejléce legyen a szokásos:

```
int main(int argc, char* argv[]);
```

2. feladat

1. lépés

A program tömörítetlen, 24bites bitmap képfájlba rejtett szöveges információ kibontását fogja végezni. 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. A képfájlok pixel adatterületének kitöltés (padding) mezője egyenértékűként kezelendő a színkomponens bájtokkal. (Szükség esetén ismerkedj meg az ASCII kódtábla és a 24bites bmp fájlok szerkezetével, működésével!) A program későbbi teszteléséhez hozz létre egy függvényt az alábbi fejléccel:

```
char* TestArray(int *NumCh);
```

A függvény hozzon létre dinamikus memóriafoglalással 3 pixel adatainak tárolására alkalmas tömböt! (Képzeljük azt, hogy ez egy bmp képfájl ún. pixel array bájtsorozatát tartalmazza!) A tömb minden elemének első 5 bitpozíciója egyenletes eloszlással véletlenszerűen legyen feltöltve, míg az utolsó 3 pozícióba a fenti leírás alapján kódold bitműveletek segítségével az „`abc`” konstans sztringet! A függvény a paraméterként megadott változó memóriaterületére helyezze el a kódolt kerekterek számát (3) majd térjen vissza a lefoglalt memóriaterület címével!

2. feladat

2. lépés

A programodban hozz létre egy függvényt a következő fejléccel!

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

A függvény első paramétere egy korábban megadott módon kódolt bmp fájl pixel array részét tartalmazó memóriaterület címe. A második paramétere a kódolt karakterek száma. A függvény foglaljon le a dekódolt szöveg számára pontosan elegendő memóriaterületet és ide mentse a

dekódolás során kapott karaktereket, majd szabadítsa fel a pixel adatokat tartalmazó (paraméterként megadott) memóriaterületet, végül a dekódolt szöveg címével térjen vissza! Ha a memóriafoglalás sikertelen, írass ki egy hibaüzenetet az alapértelmezett hiba felületre és a program álljon le visszaadva a 1 értéket (mint egyfajta hibakódot)!

2. feladat

3. lépés

A főprogramot módosítsd úgy, hogy az előző lépésben megírt függvény segítségével előállított tömböt dekódolja és ennek az eredményét írassa ki az alapértelmezett kimenetre! Leállás előtt szabadítsd fel a dinamikusan lefoglalt memóriaterületet!

3. feladat

1. lépés

Írj egy függvényt, amely beolvassa egy tömörítetlen, TrueColor bmp fájl tartalmát! Az alprogram fejléce legyen a következő:

```
char* ReadPixels(int f, int* NumCh);
```

Az ilyen képfájlok 3.-6. bájtjában (előjel nélküli egészként, little-endian bájtsorrenddel) szerepel a fájl mérete bájtokban megadva. A következő 4 bájt eredetileg kihasználatlan, de a felhasználandó kódolt bmp képekben ez a mező a kódolt karakterek (bájtok) számát tartalmazza. Ezután 4 bájton találjuk azt az egész értéket, amely megmondja, hogy hányadik bájton kezdődik a konkrét pixelek leírását tartalmazó pixel array (azaz azt, hogy hány bájtos a fejrész). A függvény első paramétere a binárisan olvasásra megnyitott fájl leírója, második paraméter output paraméterként működik. A függvény olvassa be a képfájl teljes pixel array tartalmát egy megfelelő méretű dinamikusan foglalt memóriaterületre és ennek a címével térjen vissza! Visszatérés előtt a második paraméterben megadott címre helyezze el a program a pixeltömbben kódolt karakterek számát! Ha a memóriafoglalás sikertelen, a program írjon ki hibaüzenetet az alapértelmezett hiba felületre, érjen véget és a 1 értéket adja vissza az operációs rendszernek!

3. feladat

2. lépés

Egészítsd ki a programodat egy karakteres felületű tallózást megvalósító függvénnyel az alábbiak szerint!

```
int BrowseForOpen();
```

A függvény írassa ki a kimenetre az aktuális felhasználó alapértelmezett könyvtárának tartalmát (a rejtett könyvtári objektumokat is)! A felhasználó ezek közül választ egyet és azt bemenetként megadja a programnak. Ha a megadott név nem fájl és nem könyvtár (pl. elgépelés eredménye), akkor a kérjen be új sztringet! Amennyiben a megadott karaktersorozat egy könyvtár neve, akkor a függvény listázza ki annak is a tartalmát és várjon újabb felhasználói válaszra! Ez addig ismétlődjön, amíg a felhasználó könyvtárat ad meg. A program kezelje a szülő könyvtárba való visszalépés lehetőségét is! Amikor a felhasználó egy reguláris fájl nevét adja meg bemenetként lépj ki ciklusból és nyisd meg az adott elérési úton elhelyezkedő, adott nevű fájlt binárisan kizárólag olvasásra! A függvény térjen vissza a megnyitás során kapott fájlleíróval!

3. feladat

3. lépés

Módosítsd a fő programegységet úgy, hogy amennyiben nincs megadva parancssori argumentum, akkor kerüljön meghívásra a tallózó függvény, viszont ha a képfájl neve a parancssori argumentum, akkor nyisd meg az binárisan olvasásra! Bármelyik eset is álljon fenn, ha a megnyitás sikertelen, akkor a program írjon ki hibaüzenetet az alapértelmezett hiba felületre, álljon le és adja vissza az 1 értéket az operációs rendszernek! Sikeres megnyitás esetén a kapott fájlleíróval hívd meg a `ReadPixels` függvényt és az ezáltal visszaadott tömb kerüljön dekódolásra az `Unwrap` függvénnyel! Használat

után közvetlenül zárd be a megnyitott fájlt! A korábban tesztelés céljából létrehozott `TestArray` függvényt töröld!

4. feladat

1. lépés

A dekódolt szöveget egy web szerverre kell továbbítani. Tételezzük fel, hogy a szöveg 1kB-nál nem hosszabb. Ehhez hozz létre egy socket programozáson alapuló függvényt az alábbi fejléccel:

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

Ennek első paramétere egy pointer, amely a Neptun azonosítódat tartalmazó sztringre mutat, második paramétere a dekódolt szöveg memóriaterületét adja meg, míg a harmadik a szöveg méretét bájtban. A függvény hozzon létre egy TCP kapcsolatot a `193.6.135.162` IPv4 című csomópont `80`-as portjával! Küldj egy http POST metódust leíró PDU-t az „`irh.inf.unideb.hu/~vargai/post.php`” URL-re „`application/x-www-form-urlencoded`” tartalomtípussal és két űrlapmező értékkel! Az egyik mező „`NeptunID`” névre hallgat és értékét a függvény első paramétere határozza meg, a másik mező neve „`PostedText`” és a második paraméter jelenti az értékét. Tehát egy konkrét esetben az alábbi tartalmat kell elküldeni a szervernek (a sorok végén CR+LF kombinációt, azaz ”`\r\n`”-t használva):

```
POST /~vargai/post.php HTTP/1.1
Host: irh.inf.unideb.hu
Content-Length: 33
Content-Type: application/x-www-form-urlencoded

NeptunID=ABC123&PostedText=Almafa
```

Ha a szervertől kapott http 1.1 válasz PDU tartalmazza a „`The message has been received.`” szöveget, akkor a függvény a kapcsolat bontása után térjen vissza 0 értékkel! Minden más esetben (ide értve a bármilyen kommunikáció során felmerülő hibát) a program térjen vissza pozitív értékkel (jelezve a hiba okát). (Böngésző segítségével a fenti URL-en meg is tekintheted a beküldött post-jaidat.)

4. feladat

2. lépés

Írd át a főprogramot úgy, hogy a dekódolt szöveget ne a kimenetre írja, hanem a `Post` függvénnyel küldje el! Ha a hívás 0 értéket ad vissza írass ki a képernyőre egy megerősítő nyugta szöveget, ha viszont nullától eltérő a visszaadott érték, akkor egy hibaüzenetet írass ki és térj vissza a hibakóddal!

5. feladat

1. lépés

Az `Unwrap` függvényben a tömb feldolgozása (dekódolás) a rendelkezésre álló összes processzormagon párhuzamosan történjen, közel azonos terheléssel!

5. feladat

2. lépés

Bővítsd a programot egy szignálkezelő eljárással, amely két szignált is képes kezelni! Fejléce:

```
void WhatToDo(int sig);
```

Ha a paraméter SIGALRM szignál érkezésére utal, akkor az eljárás írjon ki egy hibaüzenetet arra vonatkozóan, hogy a program túl sokáig futott, majd a program álljon le egy hibakóddal! Amennyiben SIGINT szignál aktiválta a kezelőt, akkor az eljárás hozzon létre egy gyermek folyamatot, amely arról tájékoztatja a felhasználót, hogy a ctrl+c billentyűkombináció jelenleg nem állítja meg a programot, majd a gyerek fejeződjön is be egy sajátmagának küldött SIGKILL szignál által! Ehhez használj forkolást és szignálküldést! A főprogramot módosítsd úgy, hogy a képfájl megnyitása után érkező

szignál esetén fusson le az elkészített szignálkezelő! Továbbá a fájl megnyitása után indíts el egy 1 másodperces időzítőt is, amit a dekódolás után állíts le!

5. feladat

3. lépés

A programban definiált alprogramokat szervezd ki egy külön saját header állományba (`.h`), amit a main függvény forráskódjában inkludálsz is!

6. feladat

Készíts (rövidített) dokumentációt a programodhoz, amely tartalmazza a következőket:

- A használandó fordítóprogram és annak szükséges kapcsolói.
- Felhasználói útmutatást a programhoz.
- A program által visszaadott értékek magyarázatát.
- Az elkészített alprogramok rövid leírását (paraméterezés, visszatérési érték).

7. feladat

Védd meg a programodat! Bizonyítsd be, hogy a te alkotásod! Válaszolj a feltett kérdésekre! Hajtsd végre a kért módosításokat!