Proiect programare concurenta si distribuita

Iulian Cimpan, Teodor Branescu, Sebastian Carabasiu, Bogdan Petri, Adrian Bordeianu

2020

Cuprins

1 Introducere

Proiectul pe care l-am implementat are rolul de a transfera imagini de la un calculator la altul asigurand prelucrarea acestora prin intermediul unor scripturi Python folosind mai multe biblioteci(ex: OpenCV).

2 Descriere

In dezvoltarea aplicatiei am avut ca obiective principale:

1. Crearea unui server in C
2. Crearea unui client in C
3. Introducerea de scripturi Python embedded in C pentru prelucrarea imaginilor

3 Tehnologii folosite

3.1 C server

Serverul are rolul de a primii fisierele de la client. In implementarea lui am folosit socket-uri si doua modalitati de transmitere a informatiilor:

1. UDP - User Datagram Protocol
2. TCP - Transmission Control Protocol

Pentru a rula serverul trebuie sa specificam potrul pe care acesta va functiona. Pentru server folosim biblioteci pentru:

1. time
2. sockets
3. types
4. etc.

Exemplu de functie folosita pentru crearea unui socket:

```
/* Functia care permite crearea unui socket si atasarea acestuia la sistem
 * Returneaza un descriptor de fisier in tabelul descriptorului de proces
 * bind permite definirea sa cu sistemul
 */
int create_server_socket (int port){
    int l;
   int sfd;
    int yes=1;

   sfd = socket(PF_INET,SOCK_STREAM,0);
   if (sfd == -1){
       perror("eroare socket.");
```

```
        return EXIT_SUCCESS;
    }
       /*SOL_SOCKET : manevreaza optiunile de la nivelul API-ului socket-ului
      *SO_REUSEADDR : pentru repornirea unui server in cazul unei opriri bruste
      * pentru a nu avea erori la crearea socket-ului
      * cazul in care sunt mai multe servere pe ac port
      */
     if(setsockopt(sfd, SOL_SOCKET, SO_REUSEADDR,&yes,sizeof(int)) == -1 ) {
        perror("Eroare setsocket.");
        exit(5);
     }

     //pregatirea adresei socket-ului de destinatie
    l=sizeof(struct sockaddr_in);
    bzero(&sock_serv,l);

    sock_serv.sin_family=AF_INET;
    sock_serv.sin_port=htons(port);
    sock_serv.sin_addr.s_addr=htonl(INADDR_ANY);

    //alocarea unei identitati socket-ului
    if(bind(sfd,(struct sockaddr*)&sock_serv,l)==-1){
       perror("Eroare bind");
       return EXIT_FAILURE;
    }


     return sfd;
}
```

3.2 C client

Clientul are rolul de a transmite imagini serverului, totodata acesta ocupandu-se si de prelucrarea acestora prin scripturi embedded Python. Pentru a rula clientul trebuie sa specificam: adresa ip de destinatie, portul serverului si fisierul de transferat. De asemenea si pentru client folosim doua modalitati de transmitere a informatiilor:

1. UDP - User Datagram Protocol
2. TCP - Transmision Control Protocol

Biblioteci folosite:

1. time
2. sockets
3. types
4. Python.h
5. etc.

Exemplun de functie folosita pentru crearea unui socket:

```
int create_client_socket (int port, char* ipaddr){
    int l;
   int sfd;

   sfd = socket(PF_INET,SOCK_STREAM,0);
   if (sfd == -1){
        perror("socket esuat");
        return EXIT_FAILURE;
   }

    //pregtirea adresei socket-ului de destinaie
   l=sizeof(struct sockaddr_in);
   bzero(&sock_serv,l);

   sock_serv.sin_family=AF_INET;
   sock_serv.sin_port=htons(port);
    if (inet_pton(AF_INET,ipaddr,&sock_serv.sin_addr)==0){
      printf("Adresa IP invalida:\n");
      return EXIT_FAILURE;
   }

    return sfd;
}
```

3.3 Scripturi Python

3.4 Rotire imagine

```
def rotate(angle = 30):
   angle = float(angle)
   image = Image.open(name)
   image.save(name)
   image = image.rotate(angle)
   image.save("{}".format(name))
```

3.5 Highpass

```
def highpass(sigma = 29):
   sigma = int(sigma)
   if(sigma%2):
      image = cv2.imread(name)
      blur = cv2.GaussianBlur(image, (sigma,sigma),0)
      image = image - blur
      image = image + 127
      cv2.imwrite(name, image)
   else:
      print("Sigma trebuie sa fie numar impar!")
      sys.exit()
```

3.6 Lowpass

```
def lowpass(sigma = 29):
   sigma = int(sigma)
   if(sigma%2):
      image = cv2.imread(name)
      image = cv2.GaussianBlur(image,(sigma,sigma),0)
      cv2.imwrite(name,image)
   else:
      print("Sigma trebuie sa fie numar impar!")
      sys.exit()
```

3.7 Spectrul de culoare HSV

```
def hsv():
   image = cv2.imread(name)
   image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
   cv2.imwrite(name,image)
```

3.8 Negativare imagine

```
def neg():
   image = cv2.imread(name)
   image = 255-image
   cv2.imwrite(name,image)
```

3.9 Transformarea imaginii in binar

```
def bnw():
   image = cv2.imread(name)
   image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
   (thresh, image) = cv2.threshold(image, 127,255,cv2.THRESH_BINARY)
   cv2.imwrite(name,image)
```

4 Directii viitoare de dezvoltare

Idei viitoare:

1. Adaugarea unui interfete grafice.
2. Adaugarea posibilitatii de prelucrare a unui flux video.
3. Usurarea experientei de utilizare.