

Sistem de procesare și transfer de imagini

Student: Oprica Andrei

Specializare: Informatică Aplicată

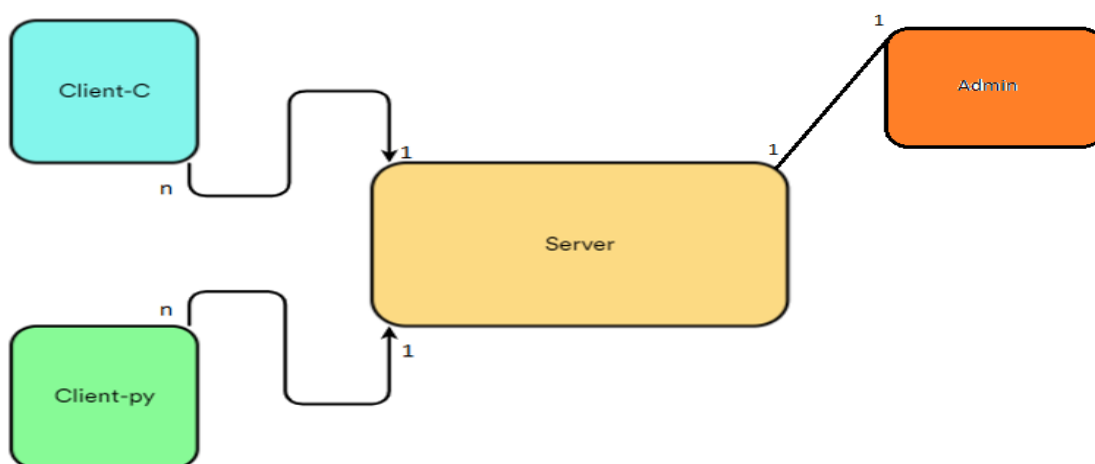
Anul: 3

Descrierea proiectului

Sistemul este format dintr-un server INET și UNIX (implementat în limbajul de programare C) ce acceptă clienți și primește de la aceștia o imagine și opțiunile acestora pentru a aplica diferite procese asupra imaginilor și a le trimite rezultatul înapoi, urmând ca clientul (nu în interfața clientului) să salveze imaginea într-un director. Pentru clienți am implementat un client INET în limbajul de programare C, pentru clienții de pe sisteme de operare bazate pe UNIX; și un client INET scris în limbajul de programare Python pentru alte sisteme de operare (cum ar fi Windows). Și un client admin în C pentru care am folosit socket UNIX.

Arhitectura sistemului

Am folosit o simplă arhitectură de tip Client-Server, în care am un server scris în C și doi clienți, unul în C și unul în Python.



Sistemul acceptă mai mulți clienți deodată, de asemenea nu contează dacă se conectează câțiva clienți de C și alții de Python.

Transferul de imagini este astfel:

- Clientul primește ca argumente o cale către o imagine și un nume pentru noua imagine.
- Clientul intră apoi într-un meniu ce îi permite să aleagă ce operație vrea să facă pe imagine.
- Apoi mărimea imaginii, imaginea și operația este trimisă către server, care primește imaginea, o salvează, o procesează și salvează rezultatul, iar apoi trimite rezultatul către client.
- Clientul primește mărimea imaginii, imaginea și salvează imaginea cu numele introdus în lista cu argumente și la final cu numărul operației.
- Apoi clientul poate să aleagă să aplice o altă procesare asupra aceleiași imagine, până când apasă o altă tastă înafară de cele din listă pentru a se deconecta de la server.
- Iar serverul ascultă în continuare după clienți.

Opțiunea utilizatorului și mărimea fișierului sunt trimise fiecare într-un pachet de 4 octeți, iar imaginea este trimisă în pachete de 1024 de octeți.

Iar clientul admin poate să interogheze serverul despre informații ca:

- Câte imagini s-au procesat;
- Câte imagini s-au procesat cu un anumit proces.

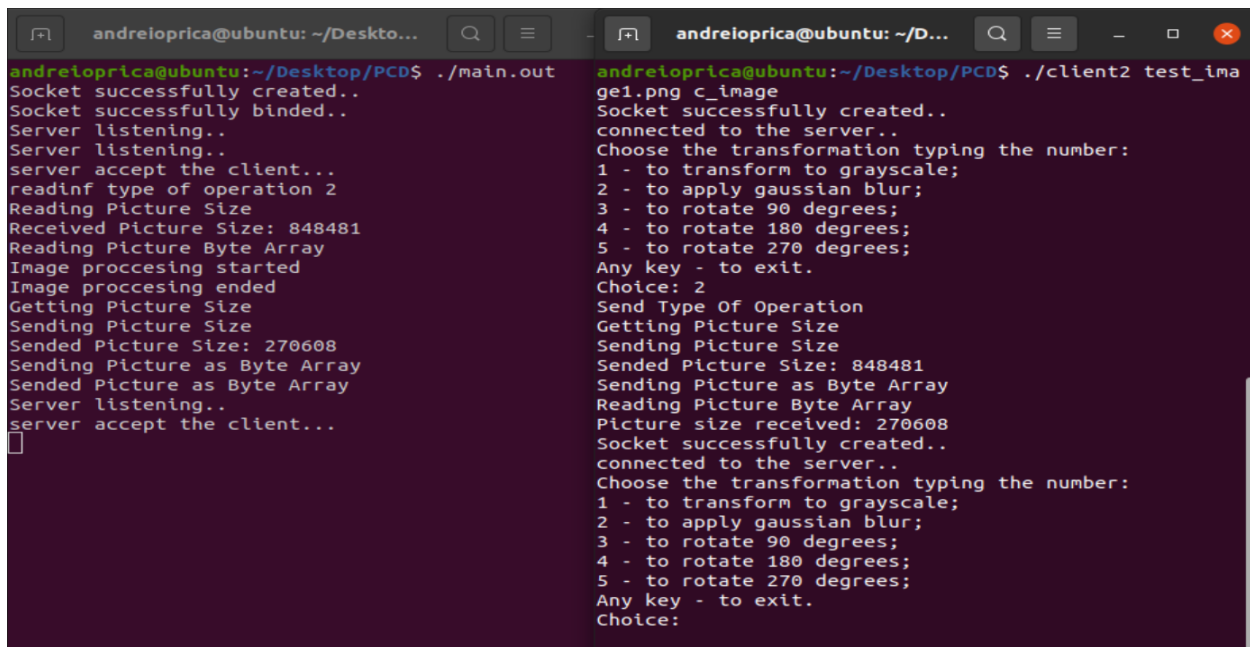
Cazurile de utilizare

Sistemul are cinci cazuri de utilizare, adică cinci funcționalități ce sunt diferite față de efectuarea transferului bidirecțional de fișiere. Aceste cinci cazuri sunt legate de procesarea de imagini. Pentru a face operații pe imagini am folosit librăria "libvips".

Cele cinci transformări de imagini sunt următoarele:

- Transformarea imaginii în alb-negru (grayscale);
- Aplicarea unui filtru de încetșoare Gaussiană (Gaussian Blur);
- Rotirea imaginii cu 90 de grade în sensul acelor de ceas;
- Rotirea imaginii cu 180 de grade în sensul acelor de ceas;
- Rotirea imaginii cu 270 de grade în sensul acelor de ceas;

Un exemplu al serverului ce ascultă, iar clientul de C face o operație ar fi următoarea:

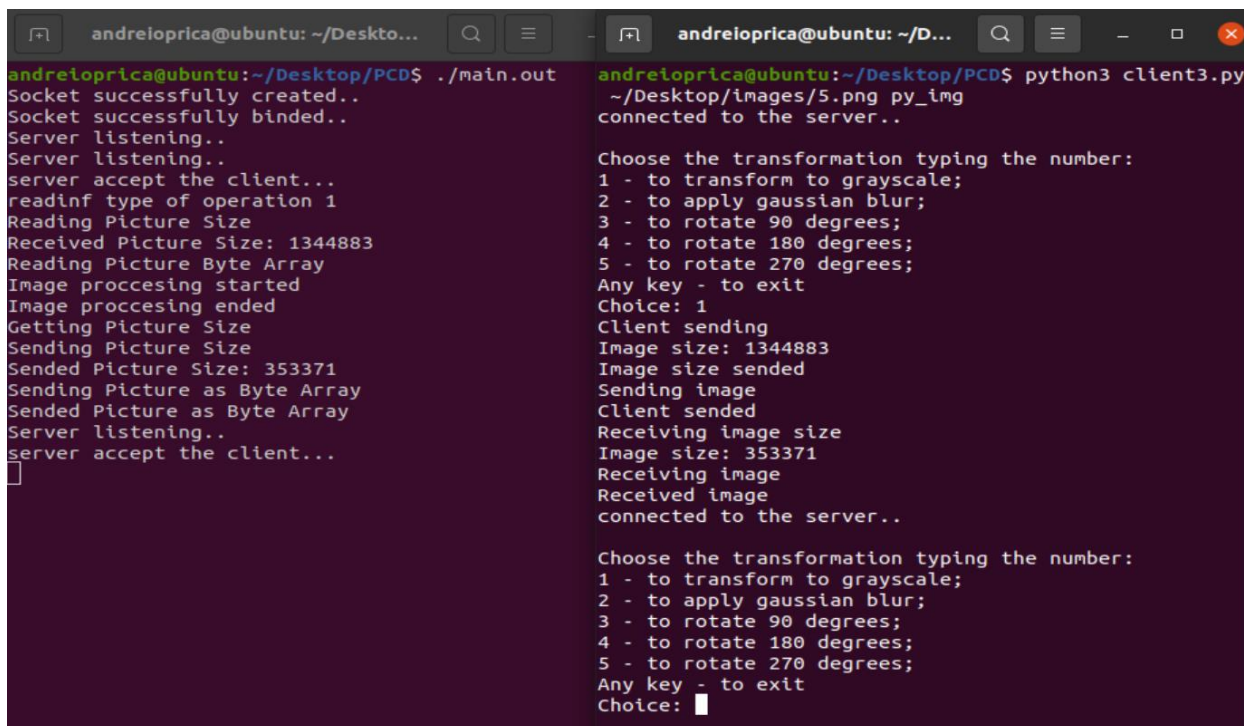


```
andreloprica@ubuntu: ~/Desktop/PCD$ ./main.out
Socket successfully created..
Socket successfully binded..
Server listening..
Server listening..
server accept the client...
readinf type of operation 2
Reading Picture Size
Received Picture Size: 848481
Reading Picture Byte Array
Image procescing started
Image procescing ended
Getting Picture Size
Sending Picture Size
Sended Picture Size: 270608
Sending Picture as Byte Array
Sended Picture as Byte Array
Server listening..
server accept the client...
[ ]

andreloprica@ubuntu: ~/Desktop/PCD$ ./client2 test_image1.png c_image
Socket successfully created..
connected to the server..
Choose the transformation typing the number:
1 - to transform to grayscale;
2 - to apply gaussian blur;
3 - to rotate 90 degrees;
4 - to rotate 180 degrees;
5 - to rotate 270 degrees;
Any key - to exit.
Choice: 2
Send Type Of Operation
Getting Picture Size
Sending Picture Size
Sended Picture Size: 848481
Sending Picture as Byte Array
Reading Picture Byte Array
Picture size received: 270608
Socket successfully created..
connected to the server..
Choose the transformation typing the number:
1 - to transform to grayscale;
2 - to apply gaussian blur;
3 - to rotate 90 degrees;
4 - to rotate 180 degrees;
5 - to rotate 270 degrees;
Any key - to exit.
Choice:
```

În acest exemplu, totul a mers bine, s-au transmis datele după ce clientul a ales o opțiune, iar după ce s-au finalizat, clientul primește opțiunea de a aplica altă procesare de imagine.

Un alt exemplu este aproape identic, dar am folosit clientul de Python pentru a exemplifica că merge.



```
andreloprica@ubuntu: ~/Desktop/PCD$ ./main.out
Socket successfully created..
Socket successfully binded..
Server listening..
Server listening..
Server accept the client...
readlnf type of operation 1
Reading Picture Size
Received Picture Size: 1344883
Reading Picture Byte Array
Image processing started
Image processing ended
Getting Picture Size
Sending Picture Size
Sended Picture Size: 353371
Sending Picture as Byte Array
Sended Picture as Byte Array
Server listening..
server accept the client...
[ ]

andreloprica@ubuntu: ~/Desktop/PCD$ python3 client3.py
~/Desktop/images/5.png py_img
connected to the server..

Choose the transformation typing the number:
1 - to transform to grayscale;
2 - to apply gaussian blur;
3 - to rotate 90 degrees;
4 - to rotate 180 degrees;
5 - to rotate 270 degrees;
Any key - to exit
Choice: 1
Client sending
Image size: 1344883
Image size sended
Sending image
Client sended
Receiving image size
Image size: 353371
Receiving image
Received image
connected to the server..

Choose the transformation typing the number:
1 - to transform to grayscale;
2 - to apply gaussian blur;
3 - to rotate 90 degrees;
4 - to rotate 180 degrees;
5 - to rotate 270 degrees;
Any key - to exit
Choice: [ ]
```

În cazul în care sunt conectați mai mulți clienți, fiecare client este servit de server secvențial în ordinea de conectare. Astfel spre exemplu dacă sunt conectați doi clienți, primul client este servit instant, în schimb dacă clientul al doilea a selectat o opțiune, imaginea i se va întoarce doar după ce si primul client a selectat opțiunea și a primit rezultatul. În acest fel se întâmplă și dacă sunt mai mulți clienți conectați, de asemenea și dacă clienții sunt conectați de pe sisteme de operare diferite (UNIX și non-UNIX).

În cazul în care imaginea nu există sau e scrisă greșit, este dat un mesaj de eroare și este oprit procesul în ambele tipuri de clienți:

```
andreioprica@ubuntu:~/Desktop/PCD$ ./client2 test_image1.pn c_image
The picture can't be open: No such file or directory
andreioprica@ubuntu:~/Desktop/PCD$ python3 client3.py ~/ges/5.png py_img
Image file do not exist.
andreioprica@ubuntu:~/Desktop/PCD$
```

Clientul admin poate să afle câte imagini s-au procesat și câte imagini s-au procesat cu un anumit proces. Mai jos voi arăta un exemplu pentru clientul admin:

```
andreioprica@ubuntu:~/Desktop/PCD$ ./client1
Admin client connected
Choose operation typing the number:
0 - Total no. of images processed;
1 - No. of images transformed to grayscale;
2 - No. of images with applied gaussian blur;
3 - No. of images rotated 90 degree;
4 - No. of images rotated 180 degree;
5 - No. of images rotated 270 degree;
Any key - to exit.
Choice: 0
9
Choose operation typing the number:
0 - Total no. of images processed;
1 - No. of images transformed to grayscale;
2 - No. of images with applied gaussian blur;
3 - No. of images rotated 90 degree;
4 - No. of images rotated 180 degree;
5 - No. of images rotated 270 degree;
Any key - to exit.
Choice: 2
2
Choose operation typing the number:
0 - Total no. of images processed;
1 - No. of images transformed to grayscale;
2 - No. of images with applied gaussian blur;
3 - No. of images rotated 90 degree;
4 - No. of images rotated 180 degree;
5 - No. of images rotated 270 degree;
Any key - to exit.
Choice: █
```

De asemenea în momentul în care clientul admin se deconectează, serverul se oprește. În cazul clienților de C și Python, la deconectarea acestora, serverul doar așteaptă clienți noi de acest tip fără să se oprească.

Cererea sistemului și utilizarea acestuia

Sistemul se poate găsi la următoarea adresă de GitHub:

https://github.com/AndreiOprica/pcd_proiect.

Sistemul pe care rulează serverul trebuie să fie UNIX, deoarece este scris în C. De asemenea sistemul trebuie să aibe instalată librăria “libvisp” ce se poate găsi la următorul link: <https://www.libvips.org/>, de unde se poate afla cum se instalează.

Pentru a funcționa sistemul are nevoie de două directoare:

- Primul este denumit: “serverOut”; și este cel în care serverul salvează imaginea primită și imaginea procesată;
- Al doilea este denumit: “Client”; și este cel în care clientul salvează imaginea procesată primită de la server.

Pentru a compila și rula serverul trebuie executate următoarele comenzi în terminal:

- Pentru compilare: `gcc -o main.out main.c `pkg-config vips --cflags --libs``
- Pentru rulare: `./main.out`

Pentru clientul admin de C trebuie urmate următoarele comenzi:

- Pentru compilare: `gcc -o client1 client1.c`
- Pentru rulare: `./client1`

Pentru clientul de C trebuie urmate următoarele comenzi:

- Pentru compilare: `gcc -o client2 client2.c`
- Pentru rulare: `./client2 path_to_image
name_processed_image`

Pentru clientul de Python trebuie executat[următoarea comandă pentru rulare:

- `python3 client3.py path_to_image name_processed_image`

Concluzii

În concluzie am creat un sistem ce procesează imagini și acceptă mai mulți clienți de pe mai multe sisteme de operare și îi rulează pe fire diferite de execuție și acceptă un client admin ce poate să vadă informații despre ce s-a procesat.

Sistemul poate face cele cinci procesări de imagini pentru imagini de tip .png deoarece este posibil ca biblioteca ce o folosesc pentru imagini să poată procesa și imagini de tip .jpg, dar clientul pune un nume, iar clientul îi adaugă direct extensia de .png, astfel cred că ar fi o problemă de fișiere să se încarce un fișier .jpg și să se salveze ca .png.