Instituto Politécnico Nacional

Escuela Superior de Cómputo

Desarrollo de un sistemas distribuidos

Diseño de sistema distribuido

Grupo: 4CM1

Integrantes:
Equipo 6

Profesor:
Coronilla Contreras Ukranio

Fecha de realización: 2 de mayo de 2020

Diseño de sistema distribuido

López Manríquez Ángel
4CM1

2 de mayo de 2020

Índice

1. Ejercicio

Con este tipo de registro y para 70 millones de votantes, ¿cuánto espacio de almacenamiento será necesario? ¿Es posible almacenarlo en su disco duro? ¿En una memoria USB? ¿Y en un archivo virtual dentro de la RAM?

Respuesta: Al ejecutar la macro `sizeof` pasandole como argumento `struct registro` vemos que ocupa 34 bytes, por tanto $34 \times 70 \times 10^6 = 2.28Gb$, por lo que si pudiese ser almacenado en cualquier unidad de almacenamiento descrita anteriormente.

2. Ejercicio

En mi servidor web se encuentra el programa *gen_registros_cel_clave_ver3.cpp* , el cual se encarga de generar n registros de votaciones con información aleatoria, pero evitando que exista un número de teléfono celular repetido. Modifique este programa para que almacene los n registros en un archivo (véase el capítulo 8 del manual de programación de sistemas LINUX), cuyo nombre se reciba como parámetro en la línea de comandos. Usando el comando time tabule los tiempos que demora en almacenar 7000, 70000,700000 y 7000000 registros en un archivo, con la computadora más rápida del equipo. Usaremos estos archivos para pruebas posteriores.

Solucion: Las funciones para salvar la estructura en un archivo son las siguientes:

```
36
37 inline unique_ptr<FILE, int(*)(FILE*) >
38 fopen_smart(const char *fname, const char *mode) {
39    return unique_ptr<FILE, int(*)(FILE*) >(fopen(fname, mode), &fclose);
40 }
41
42 inline int file_descriptor(FILE *fp) {
43    return fileno(fp);
44 }
45
46 void save_votes(const char *fname, vector<struct registro > &registers) {
47    //Aleatoriza el vector de registros e imprime el resultado
48    random_shuffle(registers.begin(), registers.end());
49    auto fp = fopen_smart(fname, "w");
50    int fd = file_descriptor(fp.get());
51    for (auto &reg1: registers) {
52       write(fd, &reg1, sizeof(reg1));
53    }
54 }
55
```

Ejecutando el siguiente script:

```
1 import os
2
```

```

fname = 'backup.txt'
for t in map(int, [70e3, 70e4, 70e5]):
    command = f'time ./a.out {t} {fname}'
    print(command)
    os.system(command)
    print('\n')


```

Tenemos el siguiente resultado por la salida estandard. Con 7 millones el archivo pesa 238 megabytes.

```
(base) aingeru@gugul:~/Documents/git/dist/12DisenoSistemaDistribuido$ python3 timing.py
time ./a.out 70000 backup.txt
0.05user 0.13system 0:00.18elapsed 99%CPU (0avgtext+0avgdata 5212maxresident)k
0inputs+4656outputs (0major+707minor)pagefaults 0swaps

time ./a.out 700000 backup.txt
0.45user 1.02system 0:01.51elapsed 97%CPU (0avgtext+0avgdata 26128maxresident)k
0inputs+46488outputs (0major+5936minor)pagefaults 0swaps

time ./a.out 7000000 backup.txt
5.67user 10.24system 0:19.27elapsed 82%CPU (0avgtext+0avgdata 235136maxresident)k
88inputs+464848outputs (0major+58229minor)pagefaults 0swaps
```

3. Ejercicio

Reutilice las clases PaqueteDatagrama y SocketDatarama para programar un cliente que lee un archivo que contiene un único registro, y lo manda en un mensaje UDP hacia el servidor de base de datos. El servidor de base de datos, después de recibir el registro lo almacena en un archivo cuyo nombre recibe como parámetro en la línea de comandos.

Solucion: Los codigos fuente para cliente y servidor son los siguientes.

```
#include "./lib/DatagramPacket.h"
#include "./lib/DatagramSocket.h"
#include "./lib/reg_util.h"


void send_register(registro r) {
    uint16_t port = 5400;
    string ip = "127.0.0.1";
    DatagramPacket pack((char *) &r, sizeof(registro), ip, port);
    DatagramSocket sock;
    sock.send(pack);
}

```

```
14 int main(int argc, char const *argv[]) {
15     const char *fname = "client.txt";
16     registro r = read_first_register(fname);
17     send_register(r);
18     puts("Registro enviado");
19     print_structure(&r, sizeof(r));
20     return 0;
21 }
```

```
1  #include "./lib/DatagramPacket.h"
2  #include "./lib/DatagramSocket.h"
3  #include "./lib/reg_util.h"
4
5
6  registro receive_register() {
7      uint16_t port = 5400;
8      registro r;
9      DatagramPacket pack((char *) &r, sizeof(registro));
10     DatagramSocket sock(port);
11     sock.receive(pack);
12     return r;
13 }
14
15 int main(int argc, char const *argv[]) {
16     const char *fname = "server.txt";
17     vector<registro> registros;
18
19     puts("Esperando registro...");
20     registro r = receive_register();
21     registros.push_back(r);
22     save_registers(fname, registros);
23
24     puts("Estructura salvada");
25     print_structure(&r, sizeof(registro));
26     return 0;
27 }
```

Imagenes del correcto funcionamiento.

```
(base) aingeru@gugul:~/Documents/git/dist/12DisenoSistemaDistribuido/ejercicio_3$ make clean
rm *.o client server
(base) aingeru@gugul:~/Documents/git/dist/12DisenoSistemaDistribuido/ejercicio_3$ make
g++ -c -g -Wall client.cpp
g++ -c -g -Wall ./lib/DatagramPacket.cpp
g++ -c -g -Wall ./lib/DatagramSocket.cpp
g++ -c -g -Wall ./lib/reg_util.cpp
g++ -g -Wall client.o DatagramPacket.o DatagramSocket.o reg_util.o -o client
g++ -c -g -Wall server.cpp
g++ -g -Wall server.o DatagramPacket.o DatagramSocket.o reg_util.o -o server
(base) aingeru@gugul:~/Documents/git/dist/12DisenoSistemaDistribuido/ejercicio_3$ ls
backup.txt  client.cpp  client.txt        DatagramSocket.o  lib       reg_util.o  server.cpp  server.txt
client      client.o    DatagramPacket.o  generator.cpp     makefile  server      server.o
(base) aingeru@gugul:~/Documents/git/dist/12DisenoSistemaDistribuido/ejercicio_3$ ./server
Esperando registro...
Estructura salvada
5504289387YEBD488687HYNASO47MOR
```

```
(base) aingeru@gugul:~/Documents/git/dist/12DisenoSistemaDistribuido/ejercicio_3$ ls
backup.txt  client.cpp  client.txt  generator.cpp  lib  makefile  server.cpp  server.txt
(base) aingeru@gugul:~/Documents/git/dist/12DisenoSistemaDistribuido/ejercicio_3$ ./client
Registro enviado
5504289387YEBD488687HYNASO47MOR
(base) aingeru@gugul:~/Documents/git/dist/12DisenoSistemaDistribuido/ejercicio_3$
```

Aqui se puede visualizar la estructura guardada en el archivo a traves del ejecutable *vim*.