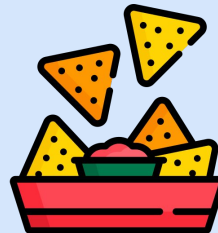


Etapa 03

Cliente en NachOS

Juan Aguilar Torres
Gabriel González Flores
Sebastián Rodríguez Tencio



Temas Abarcados

01



Servidor sin SSL

Se crea un nuevo
servidor

02



Syscalls

Se implementan los
llamados al sistema de
NachOS necesarios

03



Cliente en NachOS

Se implementa un
cliente que funciona
como un
programa de usuario
de NachOS



Servidor sin SSL

Servidor sin SSL



```
// La expresión regular para coincidir con la lista de piezas de Lego.
std::regex regex(R"((\d+)\s*(brick.*?)\s*\")");
// Iterar sobre coincidencias
std::sregex_iterator it(html_string.begin(), html_string.end(), regex);
std::sregex_iterator end;
//std::cout << "Hola displayLegos" << html << std::endl;
while (it != end) {
    // Obtener la coincidencia.
    std::smatch match = *it;
    // Extraiga la cantidad y descripción de la pieza de Lego.
    int quantity = std::stoi(match[1].str());
    total_quantity += quantity;
    std::string description = match[2].str();
    // Generar el resultado
    html = html + std::to_string(quantity) + " " + description + "\n";
    // Aumentar el iterador.
    ++it;
}
// Verificar si se encontraron piezas de Lego.
if (total_quantity == 0) {
    html = " ";
    html = html + "La figura no existe o no se encontraron piezas de lego para esta fig
} else {
    // Imprimir el total de piezas de Lego.
    html = html + "Total de piezas para armar esta figura: " + std::to_string(total_qu
}

std::string parsed = html;
```

```
int main( int argc, char ** argv ) {
    std::thread * worker;
    Socket * s1, * client;

    s1 = new Socket( 's' );

    s1->Bind( PORT );
    s1->Listen( 5 );

    for( ; ; ) {
        client = s1->Accept();
        worker = new std::thread( task, client);
        worker->detach();
    }
}
```





02

Syscalls








5

- 
- 
1. **System call write:** Escribe datos en un archivo o descriptor de archivo, como la salida estándar, para mostrar información y transferir datos entre procesos.
 2. **System call read:** Lee datos de un archivo o descriptor de archivo, como la entrada estándar, para obtener información ingresada por el usuario o recibir datos de otros procesos.
 3. **System call socket:** Crea un punto de comunicación para enviar y recibir datos entre procesos a través de una red.
 4. **System call connect:** Establece una conexión de red entre un cliente y un servidor para iniciar la comunicación y el intercambio de datos.

6

Syscalls necesarios para el cliente de NachOS

- 
-  NachOS_Write
 -  NachOS_Read
 -  NachOS_Socket
 -  NachOS_Connect



03

Cliente en NachOS



Cliente en NachOS

8

```
#include <syscall.h>

#define PORT 3141
#define BUFSIZE 512

int main( int argc, char ** argv ) {
    SpaceId newProc;
    OpenFileId input = ConsoleInput;
    OpenFileId output = ConsoleOutput;
    char prompt[8], buffer[60], receive[BUFSIZE];
    int i;

    prompt[0] = 'F';
    prompt[1] = 'i';
    prompt[2] = 'g';
    prompt[3] = 'u';
    prompt[4] = 'r';
    prompt[5] = 'a';
    prompt[6] = ':';
    prompt[7] = ' ';
```

```
Write(prompt, 8, output);
```

```
    i = 0;
    do
    {
        Read(&buffer[i], 1, input);
    } while( buffer[i++] != '\n' );
```

```
    buffer[--i] = '\0';
```

```
    int id;
    id = Socket( AF_INET_NachOS, SOCK_STREAM_NachOS );
```

```
    Connect( id, "127.0.0.1", 3141 );
```

```
    Write( buffer, 60, id );
```

```
    Read( receive, BUFSIZE, id );
```

```
    Write(receive, BUFSIZE, output);
```




Gracias!