

# Segreteria Universitaria

Avone Rita 0124/2537  
Broccoli Maria 0124/2593

Appello del 25/01/2024

## 1 Descrizione

Scrivere un'applicazione client/server parallelo per gestire gli esami universitari.

### **Segreteria:**

- Inserisce gli esami sul server dell'università (salvare in un file o conservare in memoria il dato)
- Inoltra la richiesta di prenotazione degli studenti al server universitario.
- Fornisce allo studente le date degli esami disponibili per l'esame scelto dallo studente.

### **Studente:**

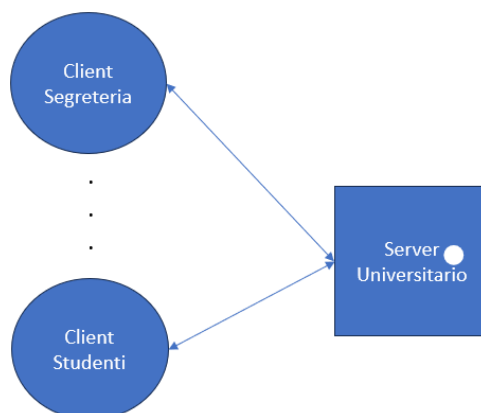
- Chiede alla segreteria se ci siano esami disponibili per un corso
- Invia una richiesta di prenotazione di un esame alla segreteria

### **Server universitario:**

- Riceve l'aggiunta di nuovi esami
- Riceve la prenotazione di un esame.

Il server universitario ad ogni richiesta di prenotazione invia alla segreteria il numero di prenotazione progressivo assegnato allo studente e la segreteria a sua volta lo inoltra allo studente.

## 2 Descrizione e schema dell'architettura



## 3 Dettagli implementativi dei client/server

### 3.1 Linguaggio di programmazione utilizzato

I codici sono implementati in C, un linguaggio di programmazione di alto livello che offre prestazioni elevate e una sintassi semplice. C è un buon linguaggio di programmazione per l'implementazione di applicazioni client-server, in quanto è efficiente e facile da usare.

### 3.2 Librerie o framework utilizzati

I codici non utilizzano librerie o framework specifici

### 3.3 Strutture dati utilizzate

I codici forniti utilizzano le seguenti strutture dati per rappresentare gli esami e le prenotazioni: - Esame: una struttura dati che contiene le informazioni sull'esame, come il nome, il corso e le date disponibili. - Prenotazione: una struttura dati che contiene le informazioni sulla prenotazione, come l'esame, la data e lo studente.

### 3.4 Protocollo di comunicazione utilizzato

I codici forniti utilizzano il protocollo TCP/IP per la comunicazione tra i client e il server. TCP/IP è un protocollo di rete affidabile che garantisce la consegna dei dati.

## 4 Parti rilevanti del codice sviluppato

### 4.1 Codice per gestire la richiesta di aggiunta di un nuovo esame nel server universitario.

```
// Funzione che gestisce la richiesta proveniente dalla segreteria
// Assume si tipo di richiesta = 0 se è di tipo 1, viene aggiunto un nuovo esame.
void gestisci_richiesta_segreteria(int socket_client, Esame *esami, int *conteggio_esami) {
    int tipo_richiesta;
    recv(socket_client, &tipo_richiesta, sizeof(tipo_richiesta), 0);

    if (tipo_richiesta == 0) { // Aggiunta esame
        // Assumiamo che la richiesta sia valida
        char nome_esame[MAX_ESAME_LENGTH];
        char data[MAX_DATA_LENGTH];
        recv(socket_client, nome_esame, sizeof(nome_esame), 0);
        recv(socket_client, &conteggio_data, sizeof(conteggio_data), 0); // Ricevo il conteggio delle date dalla segreteria
        // Ricevo la data una alla volta
        for (int i = 0; i < conteggio_data; i++) {
            int lunghezza_data; // Variabile per memorizzare la lunghezza della data
            recv(socket_client, &lunghezza_data, sizeof(lunghezza_data), 0); // Ricevo la lunghezza della data dalla segreteria

            if (lunghezza_data > 0) {
                char data[MAX_DATA_LENGTH]; // Variabile per memorizzare la data
                recv(socket_client, data, sizeof(data), 0); // Ricevo la data dalla segreteria
                data[lunghezza_data] = '\0'; // Assumiamo che la stringa ricevuta sia terminata correttamente

                // Assumiamo la data ricevuta sia strutturata appropriata (esami[conteggio_esami].date[i])
                strcpy(esami[conteggio_esami].date[i], data, MAX_DATA_LENGTH - 1);
                esami[conteggio_esami].date[i][MAX_DATA_LENGTH - 1] = '\0';
            } else {
                esami[conteggio_esami].date[i][0] = '\0'; // Se la lunghezza della data è 0, imposta la data come vuota
            }
        }

        // Aggiungo il nome alla struttura dati
        strcpy(esami[conteggio_esami].nome_esame, nome_esame);
        (*conteggio_esami)++; // Incremento il conteggio degli esami

        int risposta = 1; // Esame aggiunto con successo
        send(socket_client, &risposta, sizeof(risposta), 0);
    }
}
```

### 4.2 Codice per gestire la richiesta degli studenti per la prenotazione di un esame.

```
// Funzione che gestisce la richiesta proveniente dagli studenti
// Assume si tipo di richiesta = 0 se è di tipo 1, viene le date disponibili e viene le date dello studente.
void gestisci_richiesta_studente(int socket_client, Esame *esami, int *conteggio_esami) {
    int tipo_richiesta;
    recv(socket_client, &tipo_richiesta, sizeof(tipo_richiesta), 0);

    if (tipo_richiesta == 0) {
        char nome_esame[MAX_ESAME_LENGTH];
        if (recv(socket_client, nome_esame, sizeof(nome_esame), 0) <= 0) { // Ricevo il nome dell'esame dallo studente
            perror("Errore nella ricezione del nome dell'esame");
            close(socket_client);
            return;
        }
    }

    // Memorizzo le variabili locali
    int conteggio_data = 0;
    char data_disponibili[3][MAX_DATA_LENGTH];

    // Cerco l'esame corrispondente nella struttura degli esami
    for (int i = 0; i < conteggio_esami; i++) {
        if (strcmp(esami[i].nome_esame, nome_esame) == 0) {
            // Cerco le date disponibili per l'esame selezionato
            for (int j = 0; j < 3; j++) {
                if (strcmp(esami[i].date[j], "") != 0) {

```

```

        strncpy(date_disponibili[conteggio_date], esami[i].date[j], MAX_DATE_LENGTH - 1);
        date_disponibili[conteggio_date][MAX_DATE_LENGTH - 1] = '\0';
        conteggio_date++;
    }
    break; // Esci dal ciclo se l'esame è stato trovato
}
}

// Invia al client il conteggio delle date disponibili
if (send(socket_client, &conteggio_date, sizeof(conteggio_date), 0) <= 0) {
    perror("Errore nell'invio del conteggio delle date disponibili");
    close(socket_client);
    return;
}

// Invia le date disponibili una alla volta
for (int i = 0; i < 3; i++) {
    // Calcola la lunghezza della stringa della data
    int lunghezza_stringa = strlen(date_disponibili[i]) + 1;
    // Invia la lunghezza della stringa al client
    if (send(socket_client, &lunghezza_stringa, sizeof(lunghezza_stringa), 0) <= 0) {
        perror("Errore nell'invio della lunghezza della data disponibile");
        close(socket_client);
        return;
    }
    // Invia la data disponibile al client
    if (send(socket_client, date_disponibili[i], lunghezza_stringa, 0) <= 0) {
        perror("Errore nell'invio della data disponibile");
        close(socket_client);
        return;
    }
}

// Ricevi la scelta della data da parte dello studente
int scelta;
if (recv(socket_client, &scelta, sizeof(scelta), 0) <= 0) {
    perror("Errore nella ricezione della scelta della data");
    close(socket_client);
    return;
}

// Verifica se la scelta è valida
if (scelta >= 1 && scelta <= conteggio_date) {
    // Incrementa il numero di prenotazioni globale
    numero_prenotazione++;

    // Invia il numero di prenotazioni globale allo studente

    if (send(socket_client, &numero_prenotazione, sizeof(numero_prenotazione), 0) <= 0) {
        perror("Errore nell'invio del numero di prenotazione");
        close(socket_client);
        return;
    }
} else {
    // Se la scelta non è valida, invia la conferma di nessuna data disponibile
    int conferma_invio_date = 0;
    if (send(socket_client, &conferma_invio_date, sizeof(conferma_invio_date), 0) <= 0) {
        perror("Errore nell'invio della conferma di nessuna data disponibile");
        close(socket_client);
        return;
    }
}

// Ricevi il numero di prenotazione confermato dallo studente
int numero_prenotazione;
if (recv(socket_client, &numero_prenotazione, sizeof(numero_prenotazione), 0) <= 0) {
    printf("Errore: %d\n", errno); // Stampa il valore di errno per ottenere dettagli sull'errore
    return;
}

// Stampa il numero di prenotazione ricevuto dallo studente
printf("Numero di prenotazione ricevuto dallo studente: %d\n", numero_prenotazione);
close(socket_client); // Chiudi la connessione con lo studente
}
}

```

## 5 Manuale utente con le istruzioni su compilazione ed esecuzione

### Compilazione ed Esecuzione

#### - Server Universitario

```

[ritaavone@fedora ~]$ cd Progetto
[ritaavone@fedora Progetto]$ gcc server.c wrapper.c -o server
[ritaavone@fedora Progetto]$ ./server
Server in ascolto sulla porta 8888...

```

#### Server dopo la prenotazione

```

[ritaavone@fedora ~]$ cd Progetto
[ritaavone@fedora Progetto]$ gcc server.c wrapper.c -o server
[ritaavone@fedora Progetto]$ ./server
Server in ascolto sulla porta 8888...
Numero di prenotazione ricevuto dallo studente: 1
Numero di prenotazione ricevuto dallo studente: 2

```

#### - Segreteria

```

[ritaavone@fedora Progetto]$ gcc segreteria.c wrapper.c -o segreteria
[ritaavone@fedora Progetto]$ ./segreteria
Inserisci il nome dell'esame: Reti
Inserisci le date dell'esame una alla volta (max 3, es. 01/01/2024):
Data 1: 25/01/2024
Data 2: 15/02/2024
Data 3: 10/03/2024
Esame aggiunto con successo.
[ritaavone@fedora Progetto]$

```

#### - Studente Studente1

```
ritaavone@fedora:... x ritaavone@fedora:... x ritaavone@fedora:... x
[ritaavone@fedora Progetto]$ gcc studenti.c wrapper.c -o studente
[ritaavone@fedora Progetto]$ ./studente
Inserisci il nome dell'esame per verificare le date disponibili: Reti
Date disponibili per l'esame 'Reti':
1. 25/01/2024
2. 15/02/2024
3. 10/03/2024
Inserisci il numero corrispondente alla data scelta: 2
Prenotazione effettuata con successo. Numero di prenotazione: 1
[ritaavone@fedora Progetto]$
```

Studente2

```
ritaavone@fedora:... x ritaavone@fedora:... x ritaavone@fedora:... x
[ritaavone@fedora Progetto]$ gcc studenti.c wrapper.c -o studente
[ritaavone@fedora Progetto]$ ./studente
Inserisci il nome dell'esame per verificare le date disponibili: Reti
Date disponibili per l'esame 'Reti':
1. 25/01/2024
2. 15/02/2024
3. 10/03/2024
Inserisci il numero corrispondente alla data scelta: 2
Prenotazione effettuata con successo. Numero di prenotazione: 1
[ritaavone@fedora Progetto]$
```