Documentazione Progetto — Simulazione Mensa

Autori: Azzaro, Di Nicola, Avaro **Corso:** Sistemi Operativi — A.A. 2025/2026

---

1. Compilazione ed Esecuzione

1.1 Build

```
make
```

Il `Makefile` compila sei eseguibili nella directory `bin/`:

Eseguibile	Ruolo
`responsabile_mensa`	Processo Master — coordina l'intera simulazione
`operatore`	Operatore di stazione (Primi, Secondi, Caffè/Dolci)
`operatore_cassa`	Operatore di cassa (Cassiere)
`utente`	Processo utente della mensa
`add_users`	Utility esterna per aggiungere utenti a simulazione attiva
`communication_disorder`	Utility esterna per bloccare temporaneamente le casse

Flag di compilazione: `-Wall -Wextra -Wvla -Werror -pthread -g -D_GNU_SOURCE`, linkato con `-lrt` (POSIX timers).

Il codice sorgente è organizzato in moduli sotto `src/`:

- `src/ipc/` — Wrapper System V IPC (`shm.c`, `sem.c`, `queue.c`)
- `src/common/` — Strutture condivise e cleanup (`common.c`)
- `src/config/` — Parsing configurazione e menu (`config.c`, `menu.c`)
- `src/utils/` — Utilità generiche (random, tempo simulato, parsing)
- `src/statistics/` — Raccolta e stampa delle statistiche
- `src/programs/` — Logica dei singoli processi

1.2 Esecuzione

```
# Avvio con configurazione di default (config/config.conf)
./bin/responsabile_mensa

# Avvio con file di configurazione alternativo
./bin/responsabile_mensa config/configoverload.conf
```

Durante la simulazione, da un terminale separato:

```
# Aggiungere 15 utenti dinamicamente
./bin/add_users 15

# Provocare un Communication Disorder (blocco casse)
./bin/communication_disorder
```

Pulizia risorse IPC residue: `make clean_ipc` (equivale a `ipcrm -a`).

---

2. File di Configurazione

2.1 `config/config.conf`

Parametri chiave-valore che definiscono lo scenario simulativo. I principali:

Parametro	Descrizione
`SIM_DURATION`	Durata simulazione in giorni
`SIM_PASTO_DURATION`	Durata di un pasto in minuti simulati
`NNANOSECS`	Nanosecondi reali per minuto simulato (scala temporale)
`NOF_WORKERS`	Numero totale di operatori di stazione
`NOF_WK_SEATS_CASSA`	Numero di postazioni cassa
`NOF_USERS`	Numero iniziale di utenti
`NOF_TABLE_SEATS`	Posti a sedere totali nell'area refezione
`OVERLOAD_THRESHOLD`	Soglia di utenti non serviti per terminazione anticipata
`MAX_PORZIONI_*` / `AVG_REFILL_*`	Gestione scorte e rifornimenti
`STOP_DURATION`	Durata del blocco casse (Communication Disorder) in secondi

2.2 `config/menu.conf`

Elenco dei piatti suddivisi per categoria (primi, secondi, contorni, dolci, bevande), caricato all'avvio nella shared memory.

---

3. File Sorgente — Struttura e Ruoli

Header (`include/`)

File	Contenuto
`common.h`	Struttura principale `MainSharedMemory`, enumerazioni indici semaforici, strutture stazioni/tavoli/gruppi
`config.h`	Strutture di configurazione (`SimulationConfiguration`)
`menu.h`	Strutture del menu (`SimulationMenu`, `MenuDish`)
`message.h`	Payload dei messaggi IPC (`StationPayload`, `CashierPayload`, `ControlPayload`)
`queue.h`	Wrapper code di messaggi System V
`sem.h`	Wrapper semafori System V (P/V, barriere, wait-for-zero)
`shm.h`	Wrapper memoria condivisa System V
`statistics.h`	Strutture per statistiche giornaliere e totali
`utils.h`	Generazione random, simulazione tempo, parsing sicuro

Sorgenti del Master (`src/programs/responsabile_mensa/`)

File	Responsabilità
`responsabile_mensa.c`	Entry point: carica config, inizializza IPC, lancia figli, avvia il loop
`setup_ipc.c`	Creazione di SHM, semafori, code di messaggi
`setup_population.c`	Distribuzione operatori, fork/exec di operatori e utenti, topologia tavoli
`simulation_engine.c`	Loop giornaliero, timer, barriere, refill, gestione segnali, statistiche

---

4. Logica del Programma

4.1 Flusso Principale

1. **Inizializzazione** — Il Master (`responsabile_mensa`) carica la configurazione e il menu, alloca la Shared Memory, crea tutte le risorse IPC (semafori, code), e configura i signal handler.

2. **Spawn Popolazione** — Il Master distribuisce gli operatori tra le stazioni in base ai tempi medi di servizio, poi esegue `fork()+execl()` per creare operatori, cassieri e utenti. Ogni utente è assegnato a un gruppo casuale (1–`MAX_USERS_PER_GROUP` membri). I processi figli vengono raggruppati per tipo tramite `setpgid()` per consentire segnali broadcast.
3. **Barriera di Startup** — Tutti i processi figli attendono alla barriera iniziale. Quando tutti sono pronti, il Master apre il cancello e la simulazione comincia.
4. **Loop Giornaliero** — Per ogni giorno simulato:
   - Il Master prepara le barriere mattutine e riempie le scorte.
   - Apre il cancello mattutino: operatori e utenti iniziano a lavorare.
   - Un timer POSIX (`timer_create`) scandisce la durata del pasto; alla scadenza (`SIGALRM`), il giorno termina.
   - Un secondo timer periodico genera eventi di rifornimento (`SIGRTMIN+1`).
   - Il Master invia `SIGUSR2` (fine giorno) o `SIGTERM` (fine simulazione) a tutti i gruppi.
   - Barriera serale: tutti i figli sincronizzano la chiusura.
   - Il Master elabora eventuali richieste `add_users`, calcola sprechi, stampa il report giornaliero e controlla la soglia di overload.
5. **Terminazione** — La simulazione termina per: raggiungimento dei giorni previsti, superamento della soglia di overload, o segnale manuale (`SIGINT`). Il Master invia `SIGTERM` a tutti, attende la terminazione dei figli, e rimuove tutte le risorse IPC.

4.2 Percorso dell'Utente

Ogni utente segue un percorso sequenziale in ogni giornata:

1. **Validazione Ticket** — Se possiede un ticket (80% di probabilità), occupa un validatore (semaforo a conteggio) per la verifica, ottenendo il diritto allo sconto.
2. **Stazione Primi** — Sceglie un piatto, verifica la disponibilità (con ripiego automatico se esaurito), controlla la lunghezza della coda (soglia di pazienza), invia l'ordine via message queue e attende la risposta dall'operatore.
3. **Stazione Secondi** — Stesso meccanismo della stazione primi.
4. **Meeting Point** — Barriera di gruppo (`GROUP_SEM_PRE_CASHIER`): tutti i membri del gruppo si riuniscono prima di procedere alla cassa.
5. **Cassa** — Invia i dettagli del pasto al cassiere via message queue; il cassiere calcola l'importo (con eventuale sconto ticket del 50%) e risponde con la ricevuta.
6. **Prenotazione Tavolo** — Il leader del gruppo cerca un tavolo con posti sufficienti (mutex sui tavoli). Se non c'è posto, attende su un semaforo di

condizione finché un tavolo non si libera. Trovato il tavolo, apre il gate per i membri del gruppo (`GROUP_SEM_TABLE_GATE`).
7. **Consumazione Pasto** — Simulazione del tempo di consumo proporzionale ai piatti ordinati; alla fine, il posto viene liberato e il semaforo di condizione viene segnalato.
8. **Caffè/Dolce** — Servizio alla stazione bar.
9. **Uscita Collettiva** — Barriera di uscita (`GROUP_SEM_EXIT`): il gruppo esce insieme dalla mensa.

4.3 Operatore di Stazione

Ciclo a 3 loop annidati: - **Loop Settimanale**: attivo finché la simulazione è in corso. - **Loop Giornaliero**: sincronizzato con barriere mattina/sera. - **Loop di Lavoro**: acquisisce una postazione (semaforo `STATION_SEM_AVAILABLE_POSTS`), serve i clienti leggendo ordini dalla message queue, decrementa le porzioni in shared memory, e restituisce la risposta. Prima di ogni servizio verifica il cancello di refill (`STATION_SEM_REFILL_GATE`). Può richiedere una pausa (decisione atomica sotto mutex) purché non sia l'ultimo operatore attivo.

4.4 Utility Esterne

- **`add_users`** — Si connette alla SHM esistente tramite `ftok()`, invia una richiesta sulla coda di controllo, segnala il Master con `SIGUSR1`, attende il permesso (semaforo `MUTEX_ADD_USERS_PERMISSION`), quindi esegue fork/exec dei nuovi utenti.
- **`communication_disorder`** — Si connette alla SHM e blocca le casse incrementando `STATION_SEM_STOP_GATE` (i cassieri fanno `wait_for_zero` su questo semaforo prima di servire). Dopo la durata configurata, ripristina il semaforo.

---

5. Risorse IPC Utilizzate

L'intero sistema si basa su **IPC System V**: memoria condivisa, semafori e code di messaggi. Tutte le chiavi sono generate con `ftok("config/config.conf", 'A')` oppure con `IPC_PRIVATE` per le risorse figli-specifiche.

5.1 Memoria Condivisa (1 segmento)

Risorsa	Contenuto	Motivazione
`MainSharedMemory`	Configurazione, statistiche, menu, stato stazioni, tavoli, gruppi, registry utenti	Unico spazio di stato globale accessibile a tutti i processi. Usa un Flexible Array Member (`group_statuses[]`) per gestire i gruppi dinamicamente.

5.2 Semafori (molteplici set)

Set	Semafori	Motivazione
Barriere di sincronizzazione	6 (Startup Ready/Gate, Morning Ready/Gate, Evening Ready/Gate)	Pattern "Ping-Pong Barrier" per sincronizzare tutti i processi all'inizio/fine di ogni giornata.
Mutex globali	4 (`MUTEX_SIMULATION_STATS`, `MUTEX_SHARED_DATA`, `MUTEX_ADD_USERS_PERMISSION`, `MUTEX_TABLES`)	Protezione delle sezioni critiche in SHM: statistiche, dati condivisi, autorizzazione add_users, accesso ai tavoli.
Stazione Primi	5 (Available Posts, User Queue, Refill Gate, Refill Ack, Stop Gate)	Controllo della capacità operatori, sincronizzazione utenti-operatori, coordinamento rifornimenti.
Stazione Secondi	5 (identici)	Come sopra, per la stazione dei secondi piatti.
Stazione Caffè/Dolci	5 (identici)	Come sopra, per la stazione bar.
Cassa	5 (identici + Stop Gate per Communication Disorder)	Come le stazioni, con in più il gate di blocco utilizzato da `communication_disorder`.
Pool Gruppi	3 × N_gruppi (`PRE_CASHIER`, `TABLE_GATE`, `EXIT`)	Barriere intra-gruppo: riunione pre-cassa, attesa tavolo dal leader, uscita collettiva.

Set	Semafori	Motivazione
Ticket Validators	1 (a conteggio, init = 4)	Limita a 4 le validazioni ticket contemporanee.
Condition Tavoli	1 (segnalazione)	Pattern condvar-like: gli utenti in attesa di un tavolo dormono su `P()`; chi libera un posto fa `V()` per risvegliarli.

5.3 Code di Messaggi (5 code)

Coda	Direzione	Payload	Motivazione
Stazione Primi	Utente → Operatore (ordine), Operatore → Utente (risposta)	`StationPayload`	L'utente invia l'ordine con `mtype=MSG_TYPE_ORDER`; l'operatore risponde con `mtype=user_pid` per indirizzamento diretto.
Stazione Secondi	Identica	`StationPayload`	Come sopra.
Stazione Caffè/Dolci	Identica	`StationPayload`	Come sopra.
Cassa	Utente → Cassiere (pagamento), Cassiere → Utente (ricevuta)	`CashierPayload`	L'utente invia i dettagli del pasto; il cassiere calcola l'importo e risponde.
Coda di Controllo	`add_users` → Master	`ControlPayload`	Richieste di aggiunta dinamica utenti, elaborate dal Master a fine giornata.

Riepilogo Quantitativo

Tipo IPC	Quantità
Segmenti di Memoria Condivisa	**1**

Tipo IPC	Quantità
Set di Semafori	**10** (barriere + mutex + 4 stazioni + pool gruppi + ticket + tavoli)
Code di Messaggi	**5** (3 stazioni + cassa + controllo)

---

6. Diagramma di Sequenza

```
sequenceDiagram
    participant M as Master<br/>(responsabile_mensa)
    participant OP as Operatore<br/>(stazione)
    participant CA as Cassiere<br/>(operatore_cassa)
    participant U as Utente
    participant AU as add_users
    participant CD as communication_disorder

    Note over M: make → compilazione<br/>./bin/responsabile_mensa

    rect rgb(230, 245, 255)
    Note over M,U: Fase 1 - Inizializzazione
    M->>M: Carica config.conf e menu.conf
    M->>M: Crea SHM (shmget + shmat)
    M->>M: Crea Semafori e Code (semget, msgget)
    M->>OP: fork() + execl("operatore")
    M->>CA: fork() + execl("operatore_cassa")
    M->>U: fork() + execl("utente")
    end

    rect rgb(255, 245, 230)
    Note over M,U: Fase 2 - Startup Barrier
    OP-->>M: P(STARTUP_READY)
    CA-->>M: P(STARTUP_READY)
    U-->>M: P(STARTUP_READY)
    M->>M: wait_for_zero(STARTUP_READY)
    M->>M: open_gate(STARTUP_GATE)
    Note over OP,U: Tutti i processi sbloccati
    end

    rect rgb(230, 255, 230)
    Note over M,U: Fase 3 - Loop Giornaliero
    M->>M: Rifornimento stazioni (porzioni)
    M->>M: arm_daily_timer(SIGALRM)
    M->>M: open_gate(MORNING_GATE)
```

```
U->>U: Validazione Ticket<br/>(sem ticket_validators)
U->>OP: msgsnd(ordine primo, mtype=ORDER)
OP->>OP: Verifica porzioni (mutex SHM)
OP->>U: msgsnd(risposta, mtype=user_pid)
U->>OP: msgsnd(ordine secondo, mtype=ORDER)
OP->>U: msgsnd(risposta, mtype=user_pid)

Note over U: Meeting Point<br/>P(GROUP_SEM_PRE_CASHIER)

U->>CA: msgsnd(pagamento, mtype=ORDER)
CA->>CA: Calcolo importo + sconto ticket
CA->>U: msgsnd(ricevuta, mtype=user_pid)

U->>U: Cerca tavolo (mutex tavoli)
U->>U: Consumazione pasto (nanosleep)
U->>U: Libera posto + V(condition_tavoli)

U->>OP: msgsnd(ordine caffè, mtype=ORDER)
OP->>U: msgsnd(risposta, mtype=user_pid)

Note over U: Uscita collettiva<br/>P(GROUP_SEM_EXIT)
end

rect rgb(255, 230, 230)
Note over M: Timer scaduto (SIGALRM)
M->>OP: kill(-pgid, SIGUSR2)
M->>CA: kill(-pgid, SIGUSR2)
M->>U: kill(-pgid, SIGUSR2)
OP-->>M: P(EVENING_READY)
CA-->>M: P(EVENING_READY)
U-->>M: P(EVENING_READY)
M->>M: Statistiche giornaliere + check overload
M->>M: open_gate(EVENING_GATE)
end

rect rgb(245, 230, 255)
Note over AU,CD: Utility esterne (a simulazione attiva)
AU->>M: msgsnd(control_queue, ControlPayload)
AU->>M: kill(master_pid, SIGUSR1)
M-->>AU: V(MUTEX_ADD_USERS_PERMISSION)
AU->>U: fork() + execl("utente") × N

CD->>CD: Connessione SHM (ftok + shmget)
CD->>CA: V(STOP_GATE) - casse bloccate
Note over CA: wait_for_zero(STOP_GATE) blocca i cassieri
CD->>CA: P(STOP_GATE) - casse ripristinate
```

```
end

rect rgb(255, 240, 240)
Note over M: Fase 4 - Terminazione
M->>OP: kill(-pgid, SIGTERM)
M->>CA: kill(-pgid, SIGTERM)
M->>U: kill(-pgid, SIGTERM)
M->>M: wait() per tutti i figli
M->>M: cleanup_ipc_resources()
M->>M: Report finale + exit
end
```