Progetto SO 2025/26
Servizio Mensa 'Oasi del Golfo'

Versione non definitiva del 25 novembre 2025

Botta, Radicioni, Schifanella C.

Indice

1 Composizione gruppo di studenti

Il progetto sarà sviluppato da un gruppo, composto da 1 a **massimo 3 componenti**. Il gruppo dovrebbe essere composto da studenti dello **stesso turno**, i quali discuteranno con il docente del proprio turno. È tuttavia consentita anche la discussione del progetto di laboratorio da parte di un gruppo di studenti di turni diversi. In questo caso, **tutti** gli studenti del gruppo discuteranno con **lo stesso docente**. Esempio: Tizio (turno T1) e Caio (turno T2) decidono di fare il progetto insieme. Lo consegnano e vengono convocati dal prof. Sempronio il giorno X. Tale giorno Tizio e Caio si presentano **muniti di un computer** su cui il progetto è sicuramente funzionante e ricevono entrambi una valutazione dal Prof. Sempronio (docente del T1, anche se Caio fa parte del turno T2, il cui docente di riferimento è il prof. Calpurnio).

2 Consegna

Il progetto è costituito da:

1. il codice sorgente
2. una breve relazione che sintetizzi le scelte progettuali compiute

Il progetto si consegna compilando la seguente Google Form, cui si accede con credenziali istituzionali,

- `https://forms.gle/DMvHpL87ju4heYGA8`

la quale richiederà il caricamento di:

- progetto stesso (un unico file in formato .tgz o .zip NON .rar). Il nome del file deve essere composto dall'unione dei cognomi dei componenti del gruppo (es.: TizioCaio.zip)
- cognome, nome, matricola, email di ciascun componente del gruppo.

Dopo il caricamento del progetto, verrete convocati dal docente con cui discuterete (si veda Sezione 1 in caso di gruppo composto da studenti di turni diversi). Attenzione: **compilare una sola form per progetto** (e **non una per ogni componente del gruppo**). Una eventuale ulteriore consegna prima dell'appuntamento **annullerà la data dell'appuntamento**.

La consegna deve avvenire almeno **10 giorni prima** degli appelli scritti per dare modo al docente di pianificare i colloqui:

- se consegnate con almeno 10 giorni di anticipo rispetto alla data di un appello, il docente propone una data per la discussione entro l'appello seguente
- altrimenti, la data sarà successiva all'appello seguente.

Esempio: per avere la certezza di un appuntamento per la discussione di progetto entro l'appello scritto del 27/01/2026, si deve consegnare entro le ore 24:00 del **17/01/2026**.

3 Valutazione e validità

Il progetto descritto nel presente documento potrà essere discusso se **consegnato entro il 30 Novembre 2026**. Da Dicembre 2026 sarà discusso il progetto assegnato durante l'anno accademico 2026/27.

Tutti i membri del gruppo devono partecipare alla discussione. La valutazione del progetto è **individuale** ed espressa in 30-esimi. Durante la discussione

- verrà chiesto di illustrare il progetto;
- verrà chiesto di commentare il codice e eventualmente di apportare piccole modifiche al progetto;

- verranno proposti quesiti sul programma "Unix" del corso anche non necessariamente legati allo sviluppo del progetto.

È necessario ottenere una votazione di almeno **18** su 30 per poter essere ammessi allo scritto. In caso di superamento della discussione del progetto, la votazione conseguita consentirà di partecipare allo scritto per i **cinque appelli successivi** alla data di superamento. Non sono ammesse eccezioni o deroghe a tale periodo di validità. In caso di mancato superamento, lo studente si potrà ripresentare soltanto dopo almeno **un mese** dalla data del mancato superamento

Il voto del progetto ha un peso di 1/3 sulla votazione finale dell'insegnamento di Sistemi Operativi.

4 Pubblicazione del proprio progetto, plagio, sanzioni

Copiare altri progetti o parte di essi impedisce una corretta valutazione. Per questo motivo, gli studenti che consegnano il progetto sono consapevoli che:

- la condivisione con altri gruppi attraverso canali pubblici o privati (a titolo di esempio: google drive, canali telegram, github, etc.) di tutto o parte del progetto non è consentita fino a tutto Novembre 2026;
- la copiatura di tutto o parte del progetto non è consentita;
- eventuali frammenti di codice estratti da parti di codice visto a lezione o da altri repository pubblici devono essere opportunamente dichiarati nei commenti del codice e i candidati devono essere in grado di illustrarne il funzionamento.

Nel momento in cui uno studente non rispettasse le sopra citate norme di comportamento, dopo essere stato convocato ed aver avuto modo di illustrare la propria posizione, potrà essere oggetto delle seguenti sanzioni:

- se lo studente avrà nel frattempo superato l'esame di Sistemi Operativi anche successivamente alla data di discussione del progetto, la verbalizzazione del voto verrà annullata;
- se lo studente avrà soltanto superato la discussione del progetto ma non l'esame, la valutazione del progetto verrà annullata e lo studente non potrà accedere ad ulteriore discussione di progetto prima dei due appelli successivi alla data di evidenza di copiatura.

5 Descrizione del progetto: versione minima (voto max 24 su 30)

Si intende simulare il funzionamento di una mensa studentesca/aziendale. A tal fine sono presenti i seguenti processi e risorse: cassa di pagamento, stazioni di distribuzione cibo, tavoli, bancone del caffè, cassiere, operatori e responsabile mensa, studente/utente.

- Processo `responsabile_mensa` che gestisce la simulazione, e mantiene le *statistiche* su richieste e servizi erogati dalla mensa. Genera tutti gli altri processi del simulatore.
- Processo `operatore_stazione_primi`: eroga i primi piatti. In particolare, dovranno essere implementate le funzionalità per garantire la scelta tra almeno 2 primi piatti ogni giorno. Il tempo medio per erogare il servizio è `AVG_SRVC_PRIMI` secondi, e dovrà essere usato per generare un tempo casuale di erogazione nell'intorno $\pm 50\%$ del valore indicato.
- Processo `operatore_stazione_secondi`: eroga i secondi piatti e i contorni. In particolare, dovranno essere implementate le funzionalità per garantire almeno la scelta tra 2 secondi piatti con relativi contorni ogni giorno. Il tempo medio per erogare il servizio è `AVG_SRVC_MAIN_COURSE` secondi, e dovrà essere usato per generare un tempo casuale di erogazione nell'intorno $\pm 50\%$ del valore indicato.
- Processo `operatore_stazione_coffee`: eroga i caffè e il dolce. In particolare, dovranno essere implementate le funzionalità per garantire almeno la scelta tra 4 tipi di caffè (normale, macchiato, decaffeinato, ginseng). Il tempo medio per erogare il servizio è `AVG_SRVC_COFFEE` secondi, e dovrà essere usato per generare un tempo casuale di erogazione nell'intorno $\pm 80\%$ del valore indicato.

- Processo `operatore_cassa` (`cassiere`): gestisce i pagamenti degli utenti. Il tempo medio per erogare il servizio è `AVG_SRVC_CASSA` secondi, e dovrà essere usato per generare un tempo casuale di erogazione nell'intorno $\pm 20\%$ del valore indicato.
- Esistono `NOF_WORKERS` processi di tipo `operatore`: hanno un orario di lavoro, ed effettuano pause casuali.
- Processi di tipo `utente`. In base alla situazione delle code alle varie stazioni di erogazione cibo, l'`utente` decide in quale coda inserirsi; ottenuto il piatto, decide se procedere alla cassa o passare a un'altra stazione di distribuzione cibo, e poi recarsi alla cassa. Una volta pagati alla cassa i piatti ottenuti, mangia per un tempo proporzionale al numero di piatti acquistati, e poi torna a casa/ufficio/aula.
  - Prima di iniziare a consumare i pasti, gli `utenti` devono passare a pagare presso la `stazione_cassa`.
- Esistono `NOF_USERS` processi di tipo `utente`. Il processo `utente` decide se recarsi in mensa e sceglie cosa mangiare (si veda la descrizione del processo nella sezione 5.5).
- Esistono risorse di tipo `stazione_di_distribuzione_cibo` e `stazioni_di_refezione` (o, meno pomposamente, `tavoli`); ogni stazione è destinata a fornire un solo tipo di cibo (primo/secondo-contorno/dolce-caffè), oppure a consentire agli utenti di consumare il cibo acquistato. Esiste inoltre un altro tipo di stazione, la `stazione_cassa`.
- Ogni stazione di distribuzione è dotata di un certo numero di `postazioni`: `NOF_WK_SEATS_PRIMI`, `NOF_WK_SEATS_SECONDI`, `NOF_WK_SEATS_COFFEE`, `NOF_WK_SEATS_CASSA`.

5.1 Processo `responsabile_mensa`

Il processo `responsabile_mensa` è responsabile dell'avvio della simulazione, della creazione delle risorse di tipo `stazione_distribuzione` e `stazione_refezione`, dei processi `operatore` e `utente`, delle statistiche e della terminazione. Si noti bene che il processo `responsabile_mensa` non si occupa dell'aggiornamento delle statistiche, ma solo della loro lettura, secondo quanto indicato. All'avvio, il processo `responsabile_mensa`:

- crea `NOF_WORKERS` processi di tipo `operatore` e `operatore_cassa`, assegnandoli alle `stazioni_di_distribuzione` o alle casse, garantendo che nessuna stazione rimanga senza operatore.
- crea `NOF_USERS` processi di tipo `utente`.
- crea le 4 `stazioni_distribuzione` per primi, secondi, caffè e la cassa.
- crea `NOF_WK_SEATS_PRIMI`, `NOF_WK_SEATS_SECONDI`, `NOF_WK_SEATS_COFFEE`, `NOF_WK_SEATS_CASSA` risorse di tipo `postazione`, per la corrispondente `stazione_di_distribuzione`.
- crea `NOF_TABLE_SEATS` risorse di tipo `stazione_refezione`.

Successivamente il `responsabile_mensa` avvia la simulazione, che avrà come durata `SIM_DURATION` giorni, dove ciascun minuto è simulato dal trascorrere di `N_NANO_SECS` nanosecondi.

La simulazione deve cominciare solamente quando tutti i processi cassiere, operatore e utente sono stati creati e hanno terminato la fase di inizializzazione.

Alla fine di ogni giornata, il processo `responsabile_mensa` dovrà stampare le statistiche totali e quelle della giornata, che comprendono:

- il numero di utenti serviti totali nella simulazione;
- il numero di utenti non serviti totali nella simulazione;
- il numero di utenti serviti in media al giorno;
- il numero di utenti non serviti in media al giorno;
- il numero di piatti distribuiti totali e per tipo (primo, secondo-contorno, dolce-caffè) nella simulazione;

- il numero di piatti avanzati totali e per tipo (primo, secondo-contorno, dolce-caffè) nella simulazione;
- il numero di piatti distribuiti in media (totali e per tipo) al giorno;
- il numero di piatti avanzati in media (totali e per tipo) al giorno;
- il tempo medio (complessivo e su ciascuna stazione) di attesa degli utenti nella simulazione;
- il tempo medio di attesa (complessivo e su ciascuna stazione) degli utenti nella giornata;
- il numero di operatori attivi durante la giornata. Un operatore si considera attivo se è riuscito ad accedere ad una postazione;
- il numero di operatori attivi durante la simulazione;
- il numero medio di pause effettuate nella giornata e il totale di pause effettuate durante la simulazione;
- il ricavo giornaliero;
- il ricavo totale su tutta la simulazione e medio per ciascuna giornata (a fine simulazione).

5.2 Risorse di tipo `stazione`

All'inizio della giornata, gli operatori sono associati a una stazione (di tipo distribuzione o cassa) dal `responsabile_mensa` che assegna sempre almeno un operatore a ciascuna stazione, e poi utilizza come criterio preferenziale quello di assegnare più operatori a mansioni il cui tempo medio di svolgimento (`AVG_SRVC_*`) è più alto. La politica di associazione operatore-servizio è definita dal progettista, e deve essere applicata all'inizio di ogni giornata.

Le stazioni di refezione dei primi e dei secondi vengono inizializzate –all'inizio di ogni giorno– con un numero pari a `AVG_REFILL_PRIMI` e `AVG_REFILL_SECONDI` di porzioni per ciascun tipo di piatto; e poi alimentate, ogni 10 minuti, fino ad un massimo numero di porzioni `MAX_PORZIONI_PRIMI` e `MAX_PORZIONI_SECONDI`, uguale per tutti i tipi di piatti.

La stazione dolce e caffè ha invece una quantità illimitata di porzioni.

5.3 Processo `operatore`

All'inizio di ogni giornata lavorativa, l'`operatore`:

- compete con gli altri operatori per il posto presso la stazione a cui il `responsabile_mensa` l'ha assegnato;
- se ne trova uno, comincia il proprio turno che terminerà alla fine della giornata lavorativa;
- con un massimo di `NOF_PAUSE` volte durante ciascuna giornata, l'operatore può decidere (secondo un criterio scelto dal programmatore) di interrompere il servizio per poi riprendere il proprio compito dopo la pausa. Non tutti gli operatori possono andare in pausa contemporaneamente: non è consentito agli operatori lasciare la stazione non presidiata (almeno un operatore deve sempre essere attivo). In caso un operatore decida di mettersi in pausa:
  - termina di servire il cliente che stava servendo;
  - lascia libera la stazione occupata;
  - aggiorna le statistiche.

Il processo `operatore` che al suo arrivo non trova una postazione libera resta in attesa che una postazione si liberi (per una pausa di un altro operatore).

5.4 Processo `operatore_cassa`

Il processo `cassiere` è responsabile dei pagamenti: gli utenti che si presentano alla cassa indicano quali piatti hanno scelto e se vogliono il caffè, e il cassiere aggiorna il ricavo totale. Il costo di ciascun primo è `PRICE_PRIMI`, il costo di ciascun secondo è `PRICE_SECONDI`, e il prezzo del caffè è `PRICE_COFFEE`.

5.5 Processo `utente`

I processi di tipo `utente` si recano presso la mensa. Più in dettaglio, ogni giorno ogni processo utente:

- Stabilisce il menu che vuole acquistare (secondo un criterio stabilito dal progettista); ad esempio legge il file di configurazione `menu.txt` in cui sono elencate diverse alternative, e contiene almeno due primi e due secondi, oltre a dolce e caffè. Il file di configurazione deve rispettare una sintassi definita dal progettista, e la composizione del menu potrà essere modificata in sede di esame;
- In base alla composizione del menu scelto –che deve includere almeno un primo, un secondo ed eventualmente il dolce-caffè– si reca alle stazioni che forniscono i prodotti desiderati;
- Attende il proprio turno e l'erogazione del piatto presso ciascuna stazione di distribuzione;
- Al momento in cui l'utente sta per essere servito può capitare che il piatto desiderato sia terminato: in questo caso l'utente sceglierà un altro piatto dello stesso tipo (primi o secondi). Se tutti i piatti di quel tipo sono terminati, mangerà solo secondo o solo primo. Se tutti i piatti di entrambi i tipi sono terminati, desiste ed esce;
- Dopo avere ottenuto i piatti scelti, passa in cassa e paga;
- Si siede in una `stazione_di_refezione` dove consuma il pasto acquistato. All'interno della mensa esiste un numero di posti complessivo `NOF_TABLE_SEATS`: per potersi sedere a un tavolo, l'`utente` deve verificare che ci sia almeno un posto libero; altrimenti attende che si liberi un posto.

Se al termine della giornata l'utente si trova ancora in coda, abbandona la mensa rinunciando al pasto. Il numero di utenti che rinunciano è uno dei parametri da monitorare.

5.6 Terminazione

La simulazione termina in una delle seguenti circorstanze:

timeout raggiungimento della durata impostata a `SIM_DURATION` giorni;

overload numero di utenti in attesa al termine della giornata maggiore del valore `OVERLOAD_THRESHOLD`.

Il gruppo di studenti deve produrre e consegnare, nell'archivio contenente il progetto, diverse configurazioni (file `config_timeout.conf` e `config_overload.conf`) in grado di generare la terminazione nei casi sopra descritti.

Al termine della simulazione, l'output del programma deve riportare anche la causa di terminazione e le statistiche finali.

6 Descrizione del progetto: versione "completa" (max 30)

In questa versione:

- Attraverso un nuovo eseguibile invocabile da linea di comando si crea un processo `communication_disorder` che interrompe per un tempo `STOP_DURATION` secondi il funzionamento dei pagamenti automatici, bloccando il funzionamento della `stazione_cassa`;
- Attraverso un nuovo eseguibile invocabile da linea di comando, deve essere possibile aggiungere alla simulazione altri `N_NEW_USERS` processi utente oltre a quelli inizialmente generati dal `responsabile_mensa`;
- Tutte le statistiche prodotte devono anche essere salvate in un file testo di tipo csv, in modo da poter essere utilizzate per una analisi futura;
- Vengono distinti due tipi di utente: `utente_w_ticket` and `utente_w/o_ticket`. Le due figure sono simili, ma gli utenti con ticket hanno diritto a uno sconto, sono molto più numerosi di quelli senza ticket (l'80% del totale degli utenti), ma fanno una fila per la lettura del ticket. Il progettista deve quindi individuare degli strumenti per descrivere questa ulteriore caratteristica e dar conto di questa variante (illustrando la soluzione proposta all'interno della relazione);

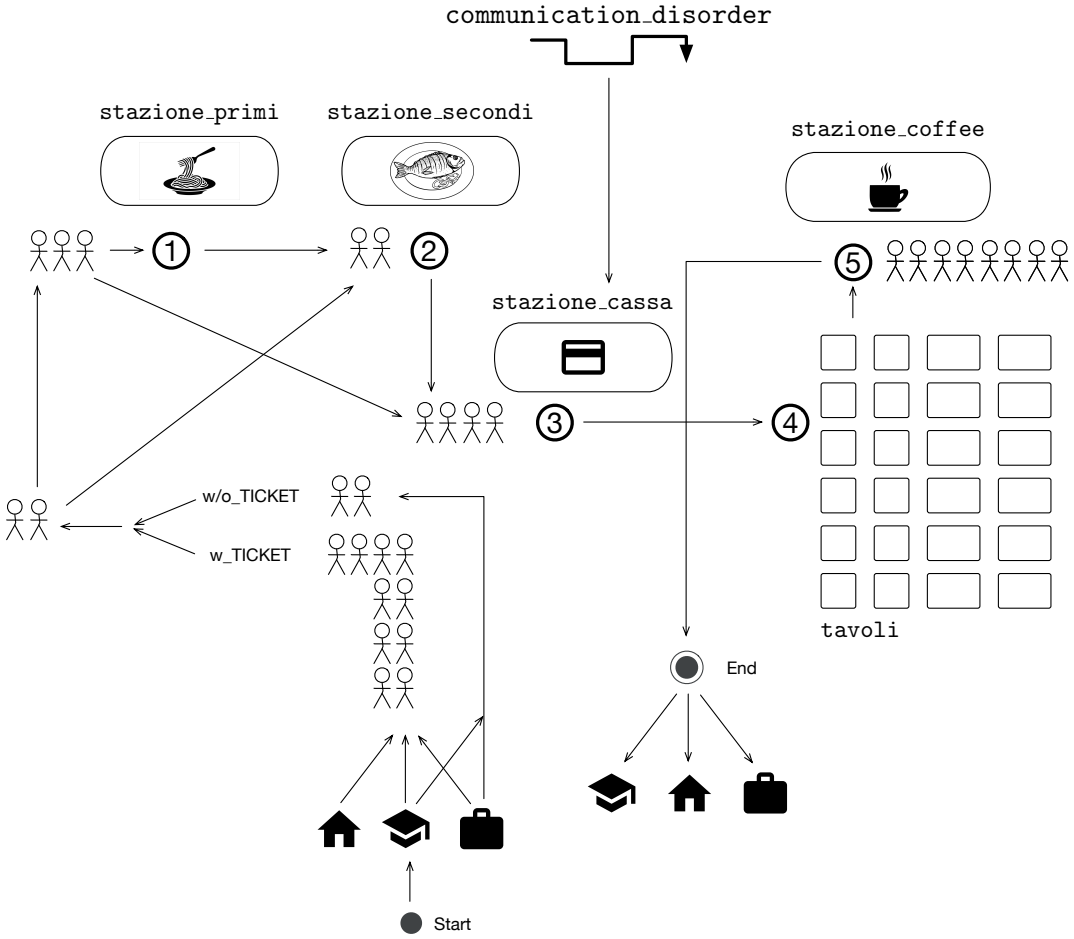

Figura 1: Schema del funzionamento della mensa.

- Gli utenti possono decidere di pranzare insieme: per mangiare insieme devono andare a pagare solo dopo essersi riforniti di tutti i piatti che desiderano acquistare. Nell'arco della simulazione ciascun gruppo è costituito (attraverso l'estrazione di un numero casuale) da un numero compreso tra uno e `MAX_USERS_PER_GROUP` utenti. L'associazione tra utenti e gruppi è determinata al momento della creazione dei processi da parte del `responsabile_mensa`;
- A differenza della versione minima, il tempo di refill dei primi e dei secondi è `AVG_REFILL_TIME`, e dovrà essere usato per generare un tempo casuale nell'intorno $\pm 20\%$ del valore indicato.

Lo schema complessivo dell'interazione è illustrato in Figura 1.

7 Configurazione

Tutti i parametri di configurazione sono letti a **tempo di esecuzione**, da file o da variabili di ambiente. Quindi, un cambiamento dei parametri non deve determinare una nuova compilazione dei sorgenti (non è consentito inserire i parametri uno alla volta da terminale una volta avviata la simulazione). Il formato del file di configurazione è a scelta del progettista.

8 Requisiti implementativi

Il progetto (sia in versione "minima" che "completa") deve

- evitare l'attesa attiva
- utilizzare almeno memoria condivisa, semafori e un meccanismo di comunicazione fra processi a scelta fra code di messaggi o pipe,
- essere realizzato sfruttando le tecniche di divisione in moduli del codice (per esempio, i vari processi devono essere lanciati da eseguibili diversi con `execve(...)`),
- essere compilato mediante l'utilizzo dell'utility `make`
- massimizzare il grado di concorrenza fra processi
- deallocare le risorse IPC che sono state allocate dai processi al termine della simulazione
- essere compilato con almeno le seguenti opzioni di compilazione:

  ```
  gcc -Wvla -Wextra -Werror
  ```

- poter eseguire correttamente su una macchina (virtuale o fisica) che presenta parallelismo (due o più processori).

Per i motivi introdotti a lezione, ricordarsi di definire la macro `_GNU_SOURCE` o compilare il progetto con il flag `-D_GNU_SOURCE`.