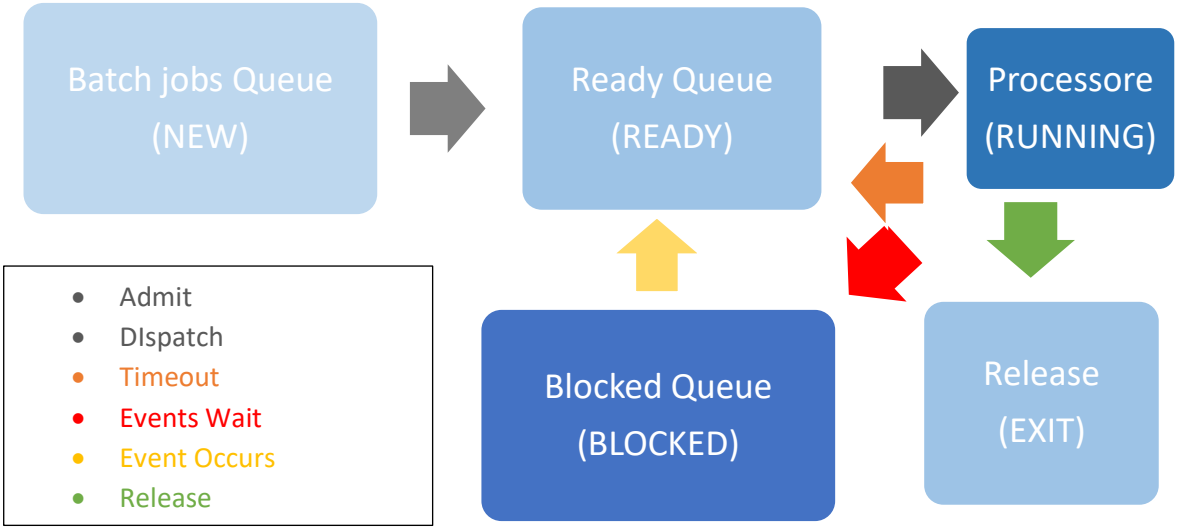
Stefano Pirastru 3942974

SCHEDULER SIMULATOR

Esame di Calcolatori Elettronici Ingegneria Informatica AA 2017 / 2018

- **Descrizione del funzionamento del simulatore:** Macchina a Stati (5 States Model)i:



Il Programma **simulatore** legge da file i Task e le istruzioni, li inserisce in una coda alla fine della lettura del file. Tutti i tasks sono nello stato iniziale new.

Vengono Istanziati due processi (**figli**) tramite fork, che si aggungono al processo creatore (**padre**).

- Uno dei problemi riscontrato nel periodo di sviluppo è stato: Il processo padre deve aspettare la terminazione dei processi figli con la funzione **Wait()** per evitare che essi diventino **Orfani** e non poter più essere terminati. Vengono 'adottati dal processo con pid 1. Ho indagato su chi fosse quale processo usando **getpid()** e **getppid().**

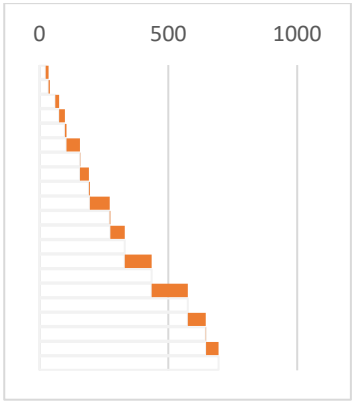
Ogni processo chiama un algoritmo di schedulazione.
Questo processo genera due thread che sono concorrenti nell'accesso alle risorse.
Ci sono tre liste in gioco: una coda di tasks in stato **new,** una coda di tasks in stato **ready** e l'ultima che contiene i tasks **blocked.**
Ho utilizzato dei **mutex** inclusi nella struct **wrapper,** che passo come argomento alla funzione routine dei thread**.** Ho preferito utilizzare la funzione di **pthread_mutex_trylock()** per essere sicuro di **non entrare** nell'area critica delle strutture dati.

- Oltre alle strutture stesse, è stato necessario applicare i mutex anche alle strutture di accesso dei dati, per evitare le concorrenze in scrittura, lettura e cancellazione o l'inconsistenza del dato.
- **I mutex** generano molte situazioni di deadlock perché il dispatcher reale del Sistema Operativo **potrebbe schedulare** i thread appena dopo che hanno chiesto un **lock** ed impedendo all'altro thread di accedere alle liste, entrando in un **loop infinito** di attesa**.**

I tasks in status exit vengono deallocati con una **free()**. Anche questa partecipa alla concorrenza.

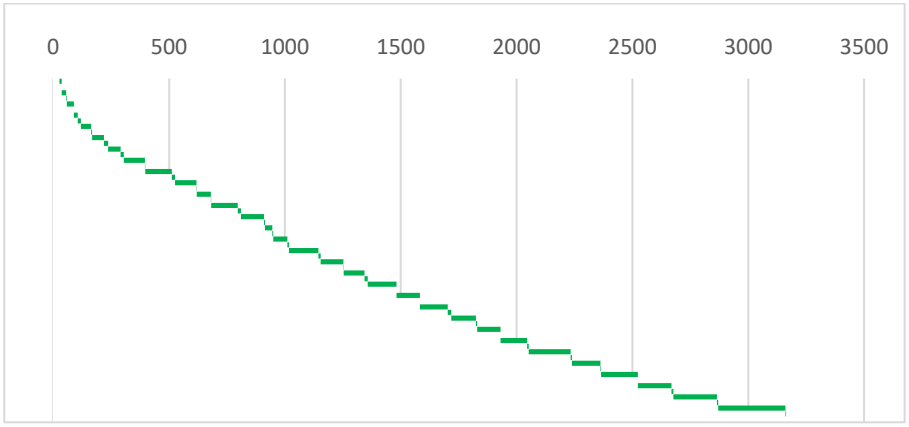
Stefano Pirastru 3942974

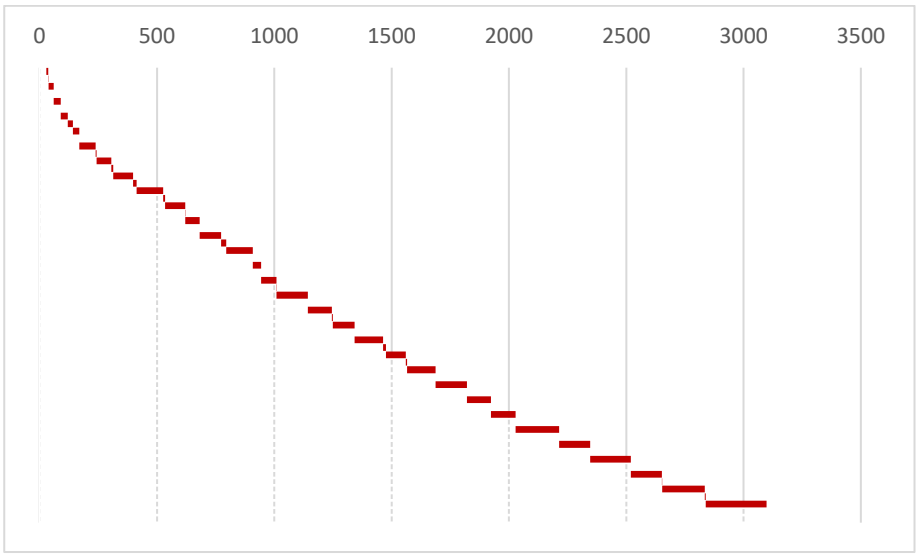
Leggendo i risultati dei primi tre tasks dal file generato dai log dei cambi di stato degli stessi ho effettuato una analisi sui tempi di completamento dei tasks con il **diagramma di Gantt**. La prima riga di ogni task è il suo tempo di arrivo. La successiva rappresenta un periodo di stato di blocco.



Differenza sulla esecuzione del task0, task1 e task2 In scala.

Nell'Asse x è definito l'andamento dei clock Ticks.





In alternativa a questa esecuzione **First Come First Served,** si può assegnare un **quanto** di tempo di esecuzione. Dopo questa finestra temporale il task non terminato passa dallo stato **running** a quello **ready**. Nel prossimo foglio descriverò i files coinvolti e il ruolo che essi hanno avuto in una simulazione, secondo gli standard **UML**.

Stefano Pirastru 3942974

- **Actors: Uml diagram**