SCHEDULER SRJF

Descrizione

Il progetto che ho sviluppato implementa uno scheduler con politica *"shortest remaining job first"*. Questo programma si sviluppa in 2 versioni, una *automatica* ed una *da-file*. Queste due versioni si differenziano per come viene generata la lista dei processi in arrivo, e come si può dedurre dal nome delle versioni, la versione *automatica* genererà la lista dei processi in arrivo con una funzione di randomizzazione, mentre la versione *da-file* genererà la lista dei processi in arrivo leggendo i dati da un file. Una volta creata la lista dei processi inizia lo scheduling vero e proprio. Ci sono 2 cicli principali, il primo pari ad un numero di volte determinato da una costante predefinita. Ad ogni iterazione un ciclo interno, controllato dal fatto che il primo elemento della lista dei processi in arrivo abbia il tempo di arrivo uguale al tempo del ciclo più esterno, prende il primo elemento della lista dei processi in arrivo, controlla se il tipo di risorsa richiesta sia la CPU o I/O, nel secondo caso lo inserisce nella lista dei processi in I/O, altrimenti controlla se il tempo di job richiesto è minore del processo in running, in caso positivo il nuovo processo entra in running e quello precedentemente in running nella lista dei processi in ready, invece in caso il processo appena arrivato allo stesso tempo rimanente del processo in running, si controlla il pid di entrambi, e solo se il pid del processo in arrivo è minore del processo in running si ha lo swapping tra i 2 processi. Nel caso il processo in arrivo non viene messo in running, viene inserito nella lista di ready. Alla fine di questo ciclo interno, si decrementa di un'unità il tempo rimanente di job del processo in running e di tutti i processi in I/O, e si controlla se il processo in running o qualcuno dei processi in I/O hanno terminato, e quelli che hanno terminato si inseriscono nella lista di waiting. Per tutti i processi nella lista di waiting viene generata, in modo randomico, una nuova richiesta di risorsa, sia per il tipo (CPU o I/O) sia per il tempo. Una volta terminato il primo ciclo principale si entra in un secondo ciclo che non si esaudisce fino a quando ci sono elementi nella lista di arrivo. Il lavoro all'interno di questo ciclo è identico al precedente, con l'unica differenza che non vengono generate nuove richieste di risorse, così da poter terminare il programma.

Comandi

Per **generare** l'eseguibile, da terminale dalla cartella *progetto_Sistemi_Operativi_2018-19/scheduler_simulator/SRJF* inviare il comando `make`.
Per **eseguire tutte le versioni** dell'eseguibile, da terminale dalla cartella *progetto_Sistemi_Operativi_2018-19/scheduler_simulator/SRJF* inviare il comando `make runAll`.
Per **eseguire una volta la versione *automatica*** dell'eseguibile, da terminale dalla cartella *progetto_Sistemi_Operativi_2018-19/scheduler_simulator/SRJF* inviare il comando `make run_auto`.
Per **eseguire una volta la versione *da-file*** dell'eseguibile, da terminale dalla cartella *progetto_Sistemi_Operativi_2018-19/scheduler_simulator/SRJF* inviare il comando `make run_file`.

SCHEDULER SRJF

Per **eseguire con vari parametri la versione *automatica*** dell'eseguibile, da terminale dalla cartella *progetto_Sistemi_Operativi_2018-19/scheduler_simulator/SRJF* inviare il comando `make go_auto`.
Per **eseguire con vari parametri la versione *da-file*** dell'eseguibile, da terminale dalla cartella *progetto_Sistemi_Operativi_2018-19/scheduler_simulator/SRJF* inviare il comando `make go_file`.
Per **testare con valgrind la versione *automatica*** dell'eseguibile, da terminale dalla cartella *progetto_Sistemi_Operativi_2018-19/scheduler_simulator/SRJF* inviare il comando `make test_auto`.
Per **testare con valgrind la versione *da-file*** dell'eseguibile, da terminale dalla cartella *progetto_Sistemi_Operativi_2018-19/scheduler_simulator/SRJF* inviare il comando `make test_file`.
Per **misurare il tempo impiegato dalla versione *automatica*** dell'eseguibile, da terminale dalla cartella *progetto_Sistemi_Operativi_2018-19/scheduler_simulator/SRJF* inviare il comando `make timing_auto`.
Per **misurare il tempo impiegato dalla versione *da-file*** dell'eseguibile, da terminale dalla cartella *progetto_Sistemi_Operativi_2018-19/scheduler_simulator/SRJF* inviare il comando `make timing_file`.
Per **rimuovere l'eseguibile e tutti i file .txt generati,** da terminale dalla cartella *progetto_Sistemi_Operativi_2018-19/scheduler_simulator/SRJF* inviare il comando `make clean`.

Parametri

Alle due versioni dello scheduler così generate si possono passare vari parametri; la versione automatica accetta le seguenti modalità di passaggio di parametri:

```
./srjf_sim_auto <fileOut>
```

oppure

```
./srjf_sim_auto <maxNumProc> <maxTime> <maxDurationProc>
```

oppure

```
./srjf_sim_auto <maxNumProc> <maxTime> <maxDurationProc>
<fileOut>
```

La versione *da-file* invece accetta le seguenti modalità di passaggio di parametri:

```
./srjf_sim_file <fileIn>
```

oppure

```
./srjf_sim_file <fileIn> <fileOut>
```

oppure

```
./srjf_sim_file <maxTime> <maxDurationProc> <fileIn>
```