Elaborato 2: Processi e IPC.

Consegna entro: 27/6/2014.

Testo dell'elaborato

Si vuole realizzare un programma in C che utilizzi le system call (IPC), ove possibile, per implementare lo scheletro di un simulatore multiprocessore con memoria RAM. Il progetto deve essere commentato in formato Doxygen, e corredato da uno script configurazione per tale tool e da uno script Makefile per la sua compilazione. Inoltre si devono allegare al progetto anche eventuali file di supporto.

I file di traccia avranno la seguente struttura:

- S <*num*>: indica una operazione di sleep pari a *num* secondi. Serve per simulare una computazione della durata di *num* secondi.
- W <*num*> <*pos*>: indica una operazione di scrittura in memoria RAM, del valore intero *num*, all'indirizzo *pos*.
- R <*pos*>: indica una operazione di lettura di numero intero dalla memoria RAM, all'indirizzo *pos*.

Ad esempio, alcuni file di configurazione potrebbero essere:

File 1	File 2	File 3	File 4
S 1 W 2 4 S 3 R 7	S 2 R 4 S 1 W 2 7	W 1 2 R 2 S 1 R 1	W 3 2 W 20 1 R 2 S 2

Il programma funzionera' nel modo seguente.

Il programma prendera' due interi dall'utente, *numCpu* e *ramDim*. Gli interi possono essere passati da linea di comando oppure in modo interattivo.

Il programma crea quindi:

- Tutte le eventuali risorse che saranno necessarie per la simulazione (es. semafori e memoria condivisa).
- Il processo che simula la RAM.
- Un numero di processi pari a *numCpu* per simulare i processori.

Attende quindi la fine della simulazione aspettando la terminazione dei processi che simulano i processori, fa terminare il processo che simula la RAM, libera le risorse allocate, e termina.

Il processo che simula la RAM eseguira' in questo modo:

1. Alloca un array di interi, pari a *ramDim*, per simulare la RAM, e lo inizializza a zero.
2. Entra in un loop:

1. Attende che arrivi una richiesta. Tutte le richieste saranno passate tramite memoria condivisa.
2. Se la richiesta e' di tipo W (scrittura), scrive il valore *num* all'interno della cella *pos* dell'array precedentemente allocato.
3. Se la richiesta e' di tipo R (lettura), legge la cella *pos* dell'array e ne scrive il valore in memoria condivisa.

Ogni processo che simula un processore eseguira' in questo modo:

1. Legge il file di traccia che deve simulare. Il file sara' nominato *cpu_<id>.txt*, dove *id* e' il numero del processore considerato (a partire da 1).
2. Entra in un loop per simulare la traccia:
    1. Se l'operazione e' di tipo S (sleep), si addormenta tramite una sleep per un numero di secondi pari a *num*.
    2. Se l'operazione e' di tipo W (scrittura), scrive in memoria condivisa la richiesta, e segnala al processo che simula la RAM che una nuova richiesta e' disponibile.
    3. Se l'operazione e' di tipo R (lettura), scrive la richiesta in memoria condivisa, segnala al processo che simula la RAM che una nuova richiesta e' disponibile, ed attende la risposta (cioe' attende il valore che ha richiesto di leggere).
3. Quando ha finito di simulare tutta la traccia, termina.

Quando i processi devono attendere, non devono fare attese attive: si devono bloccare su semafori. Allo stesso modo, la “segnalazione” di una richiesta o che un dato e' pronto va sempre fatta tramite semafori.

Si aggiungano ai processi delle stampe a video per poter seguire l'esecuzione delle tracce.

Per ogni chiamata ad una system call, si deve controllare che tale funzione abbia successo.

Ove possibile, si devono usare le system call al posto delle equivalenti chiamate a funzioni di libreria.

Tutte le stampe a video, le letture e le scritture su file devono avvenire tramite system call (quindi ad esempio non si possono utilizzare *printf*, *fprintf*, *scanf*, *fscanf*, *perror*).

FAQ

1. *E’ possibile inserire il codice su file separati?*

   Questa decisione e' lasciata allo studente, che puo' scegliere il modo piu' opportuno. L'importante e' che il codice sia ben strutturato e leggibile.

2. *Si possono usare funzioni quali la printf, fscanf, etc.?*

   No. L'elaborato richiede di usare le system call ove possibile, e tali funzioni sono rimpiazzabili tramite le system call *open*, *write*, etc. In generale, si cerchi di utilizzare il piu' possibile le system call. Invece, si possono usare funzioni quali la *sprintf*, perche' non ha una system call equivalente.

3. *Alcune cose non sono ben specificate nell'elaborato. Cosa faccio?*

   Alcune cose non sono specificate apposta per lasciare liberta' agli studenti di implementarle come preferiscono. In caso di dubbi, si possono comunque contattare i docenti per eventuali chiarimenti.

4. *Quale dei due stili presentati a lezione bisogna usare per il Makefile?*

   Lo studente puo' usare quello che preferisce. L'importante e' che sia usato uno dei due stili presentati a lezione.

N.B.: Tutto quanto non esplicitato in questo documento può essere implementato liberamente.