Prova pratica di Calcolatori Elettronici (nucleo v6.*)

C.d.L. in Ingegneria Informatica, Ordinamento DM 270

18 giugno 2020

1. Vogliamo aggiungere al nucleo il meccanismo dei *gruppi di processi*. Un qualunque processo può attivare un nuovo gruppo e altri processi possono unirsi. Un processo può poi mettersi in attesa che uno qualunque dei processi di un gruppo termini.

I gruppi sono identificati da un numero tra 0 e MAX_GRP -1. Un gruppo è vuoto se non vi appartiene nessun processo e attivo altrimenti. Al più MAX_GRP gruppi possono essere attivi in ogni istante.

Ogni processo può appartenere al più ad un gruppo alla volta e, alla creazione, non appartiene a nessun gruppo. Un processo entra a far parte di un gruppo o attivando un gruppo precedentemente vuoto (primitiva newgrp(), che restituisce l'identificatore del gruppo appena attivato) o aggiungendosi ad un gruppo già attivo (primitiva joingrp(gid), dove gid deve essere l'identificatore del gruppo). Un processo abbandona il gruppo a cui appartiene se invoca la primitiva leavegrp(), se termina/abortisce, oppure se attiva un altro gruppo invocando newgrp() (non si può invece abbandonare un gruppo invocando direttamente joingrp() su un altro gruppo).

Un qualunque processo può invocare waitgrp(gid) e sospendersi in attesa della terminazione/aborto di uno qualunque dei processi che ancora fanno parte del gruppo gid. La primitiva restituisce l'identificatore del processo terminato o abortito. Se il gruppo gid è inizialmente vuoto, o si svuota in seguito, la primitiva restituisce invece 0xFFFFFFF. Si noti che un gruppo può svuotarsi perché i processi che ne fanno parte possono abbandonarlo senza terminare o abortire. Più processi possono aver invocato waitgrp(gid) sullo stesso gruppo ed essere tutti contemporanemente in attesa. Tutti i processi in questa condizione si risveglieranno insieme e riceveranno lo stesso risultato.

Per realizzare il meccanismo aggiungiamo le seguenti primitive (abortiscono il processo in caso di errore).

- natl newgrp(): attiva un gruppo precedentemente vuoto e ne restiutisce l'identificatore. Il processo invocante entra nel nuovo gruppo e abbandona l'eventuale gruppo precedente. La primitiva restiuisce Oxffffffff e non ha altri effetti se, e solo se, ci sono già MAX_GRP gruppi attivi.
- bool joingrp(natl gid): tenta di entrare nel gruppo gid. La primitiva fallisce, e restiuisce false, se il gruppo gid non è attivo, altrimenti restituisce true. È un errore invocare la primitiva se il processo appartiene già ad un gruppo (anche se si tratta dello stesso gruppo gid), o se gid non è un identificatore valido.
- void leavegrp(): il processo abbandona il gruppo corrente torna a non appartenere ad alcun gruppo. È un errore invocare la primitiva se il processo non appartiene ad un gruppo.
- natl waitgrp(natl gid): sospende il processo in attesa che uno qualunque dei processi del gruppo gid termini/abortisca. Restiuisce l'identificatore del processo terminato/abortito, oppure Oxffffffff se il gruppo è vuoto o si svuota prima che qualunque processo del gruppo termini/abortisca. È un errore se gid non è un identificatore valido.

Modificare i file sistema.s e sistema.cpp per realizzare il meccanismo. Gestire correttamente i casi di preemption.