

# Prova pratica di Calcolatori Elettronici (nucleo v6.\*)

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

10 gennaio 2013

1. Vogliamo aggiungere al nucleo una primitiva `join()` tramite la quale un processo può sospendersi in attesa della terminazione di uno (qualunque) dei processi da lui creati tramite `activate_p()`.

Per realizzare tale primitiva introduciamo la seguente struttura:

```
struct family {
    natl nchildren;
    bool orphan;
    struct des_proc *waiting;
};
```

La struttura `family` rappresenta una “famiglia” di processi, composta da un processo “padre” e dei processi “figli”, creati dal padre tramite `activate_p()`. Il campo `nchildren` contiene il numero di figli creati e non ancora terminati. Il campo `orphan` vale `true` se e solo se il padre è terminato. Il campo `waiting` è una coda su cui il processo padre può bloccarsi in attesa della terminazione di uno dei suoi figli.

Ogni processo appartiene a due famiglie: quella in cui è un figlio e quella in cui è a sua volta padre. Quindi aggiungiamo due campi al descrittore di processo:

```
family *parent;
family *own;
```

Il campo `parent` punta alla famiglia in cui il processo a cui appartiene il descrittore è un figlio, e il campo `own` alla famiglia in cui è padre (i suoi figli punteranno alla stessa struttura, tramite il loro campo `parent`). Le strutture puntate da tali campi vengono allocate e inizializzate alla creazione del processo. Quando un processo vuole attendere che uno dei suoi figli termini si blocca sulla lista `waiting`. Quando un processo termina, controlla la lista `waiting` nella famiglia `parent` e, se necessario, risveglia il processo padre.

Poiché le strutture `family` sono condivise tra più processi (padre e figli) che possono terminare in qualsiasi ordine, dobbiamo porre particolare attenzione alla loro deallocazione. Adottiamo le seguenti regole, seguite da ogni processo alla propria terminazione:

1. se la famiglia `own` non ha figli, dealloca la struttura, altrimenti pone `orphan` a `true`;
2. decrementa `nchildren` nella famiglia `parent`; se il padre non è ancora terminato non fa altro, altrimenti dealloca la struttura se non ci sono altri figli.

**Attenzione:** per motivi tecnici alcuni processi non hanno un padre. In quel caso il puntatore `parent` deve essere inizializzato a 0 e ovviamente non vanno eseguite le azioni del punto 2.

Modificare i file `sistema.cpp` e `sistema.s` in modo da realizzare le primitive e il codice mancante.