Problema:

Un **laboratorio di Sistemi Operativi** è gestito da un professore e due tutor. Durante la lezione gli N alunni possono incontrare delle difficoltà nello svolgimento delle esercitazioni. Un alunno in difficoltà chiede aiuto al tutor col minor numero di richieste, il quale, appena libero, si reca dall’alunno. Nel caso in cui il tutor non riesca a risolvere il problema in T unità di tempo, lascia l’alunno che si rivolge al professore. Risolta la difficoltà, l’alunno prosegue l’esercitazione fino alla fine della lezione. Si supponga che la lezione inizi contemporaneamente per tutti gli alunni e che ogni alunno possa richiedere più volte l’aiuto dei tutor e del professore.

**var:**

tutor\_libero[i] : array di semafori binari (:=1) *//inizializzati a 1 perché all’inizio i tutor sono liberi*

coda\_tutor[i] : array di semafori contatore (:=N) *//inizializzati a N perché ogni tutor ha una coda di N studenti*

studente\_pronto[N] : array di semafori binari (:=0) *//ogni studente ha il proprio semaforo binario*

studente\_da\_prof[N] : array di semafori binari

problema\_risolto :

flag difficoltà : int *//false: nessuna difficoltà per lo studente; true: difficoltà per lo studente*

**Processi**: studente, assistente, professore

**void PROF\_HAND (int sign)** { *//nell’handler si libera il tutor e si risveglia il professore*

signal (tutor\_libero[i])

signal (studente\_da\_prof[N])

}

**studente ()**

begin

repeat *//per tutta la durata della lezione, controlla se ci sono problemi*

difficoltà = verifica\_difficoltà *//funzione che verifica se ci sono difficoltà nello svolgimento del compito*

if (difficoltà == true) *//ci sono difficoltà, lo studente ha bisogno di aiuto*

if trywait (coda\_tutor[1]) >= trywait (coda\_tutor[2]) { *//verifichiamo la coda più breve*

wait (coda\_tutor[1]) *//si accoda al tutor1*

wait (tutor\_libero[1]) *//attende che si liberi il tutor1*

*//si avvicina al tutor perché è arrivato il suo turno*

signal (studente\_pronto[N]) *//risveglia il tutor1*

*//comunica il problema al tutor*

wait (problema\_risolto) *//attende che il tutor1 o il prof risolvano il problema*

signal (coda\_tutor[1]) *//lascia la coda di tutor1*

*//continua il compito*

}

else {

wait (coda\_tutor[2]) *//si accoda al tutor2*

wait (tutor\_libero[2]) *//attende che si liberi il tutor2*

*//si avvicina al tutor perché è arrivato il suo turno*

signal (studente\_pronto[N]) *//risveglia il tutor2*

*//comunica il problema al tutor*

wait (problema\_risolto) *//attende che il tutor2 o il prof risolvano il problema*

signal (coda\_tutor[1]) *//lascia la coda di tutor2*

*//continua il compito*

forever

end

**tutor (i)**

begin

signal (SIGALARM, PROF\_HAND) *//installa l'handler nel tutor*

repeat *//per tutta la durata della lezione attende degli studenti in difficoltà*

wait (studente\_pronto) *//attende uno studente*

alarm (T) *//parte il timer - dopo T secondi richiama l'handler che risveglia il professore*

*//risolve il problema dello studente*

signal (problema\_risolto) *//segnala allo studente di aver risolto il problema*

signal (tutor\_libero[i]) *//risolto un problema, segnala che è dinuovo libero*

forever

end

**professore()**

begin

repeat *//ripete per tutta la durata della lezione se ci sono studenti che hanno bisogno del suo aiuto*

wait (studente\_da\_prof) *//si mette in attesa di uno studente*

*//risolvi il problema*

signal (problema\_risolto) *//segnala allo studente la risoluzione del problema per risvegliarlo*

forever

end

Problema 31-01-2011 **Laboratorio di ricerca**

Var:

struct {

codice\_personale: array di int

tipologia\_personale: array di boolean

postazioni\_A: semaforo generico (:=K)

postazioni\_B: semaforo generico (:=N-K)

personale: semaforo binario (:=0) //segnalare la presenza di personale al coordinatore

mutex: semaforo\_binario (:=0) //accesso in CS

time\_B: int //tempo trascorso per il personale\_B

} pers

Coordinatore (tipologia\_personale)

Begin

Wait (personale) //attende il personale

In base a tipologia\_personale controlla la disponibilità di posti nei K o negli N-K

Una volta controlla la disponibilità, inviamo l’esito al personale

Signal (esito)

End

Personale\_laboratorio ()

Wait (mutex) //assegnamo un ID al personale

Pers.Codice\_personale[i]=contatore

Signal (mutex)

Signal (personale) //risveglia il coordinatore

Wait (esito) //aspettiamo l’esito del controllo effettuato dal coordinatore, se questo è positivo, allora ci sono posti liberi, se è negativo non ve ne sono

Nel caso in cui, l’esito sia negativo dovremmo, se siamo personale\_A uscire, invece, se siamo personale\_B fare una wait su postazioni\_B

**Idea**

L’idea è quella di realizzare un unico processo per entrambi i tipi di personale ed affiancando a ciascuno di essi un flag che indica se appartiene alla categoria A o alla categoria B (1)

**Domande**

1. Come si fa a comunicare al coordinatore la tipologia del personale? Potrebbe essere un parametro della procedura?
2. Come gestire le due tipologie di personale? E’ conveniente fare due processi distinti oppure un unico processo? La conseguenza è il “wait” del coordinatore perché se facciamo due processi, li attenderebbe entrambi?