Laboratorio di Sistemi Operativi 2015-2016: esercitazione finale.

Hospital-OS

Il sistema da realizzare gestisce servizi di 2 tipi: *triage* e *erogazione di prestazioni* ospedaliere.

Un numero limitato (fissato in un file di configurazione; default: 10 accessi) di pazienti accede all'ospedale (*SEM*). Una volta all'interno, i pazienti accedono al triage sulla base dell'ordine di arrivo (*CODA*).

I pazienti accedono al triage lamentando un certo *sintomo* (generato a caso fra un elenco di sintomi letti da file), utilizzato dal personale del triage per valutare la gravità della patologia in questione e indirizzare il paziente in un certo reparto. L'associazione sintomo-*codice di gravità* è fissa, e letta da file: a ogni sintomo è quindi associata una certa gravità, su una scala da 1 a 10.

Dopo la valutazione effettuata presso il triage, ogni paziente viene indirizzato verso un certo reparto (deve funzionare con un *n* da 1 a 10; default: 2, *n* fissato in file di configurazione), e provvisto di un *codice di gravità* (1-10). Ogni reparto ha associato un *FIFO* a cui il triage comunica i dati del paziente e il codice di gravità.

Una volta nei reparti, a intervalli costanti i pazienti sono accolti e associati a un *numero di turno*, calcolato in modo da combinare due criteri:

- pazienti con la stessa gravità vengono visitati sulla base dell'ordine di arrivo.
- pazienti più gravi vengono serviti prima, garantendo però che i pazienti meno gravi non attendano indefinitamente.

Una volta calcolato il numero di turno, il paziente viene inserito in una *CODA* con priorità da cui viene selezionato per l'erogazione della prestazione necessaria. Dopo avere atteso un tempo casuale (corrispondendente all'erogazione della prestazione), il paziente esce dall'ospedale, effettua l'azione opportuna sul semaforo che disciplina gli ingressi, e termina.

Dopo un certo tempo (fissato in un file di configurazione; default: 20 secondi), il sistema 'chiude' (*SIGALARM*), continuando a servire i pazienti già entrati ma non accettandone altri.

Terminazione

Il sistema potrà essere terminato con segnale *SIGQUIT*; alla ricezione del segnale deve ripulire la memoria di tutte le risorse utilizzate durante il suo servizio.

Note implementative

All'interno delle varie strutture contenenti richieste e risposte ***non*** devono essere presenti array dichiarati staticamente: è cioè necessario utilizzare *sempre* l'allocazione dinamica.

Processi ausiliari

È necessario realizzare un *generatore di processi*, e predisporre un insieme di stampe a video e su file che consentano di comprendere con chiarezza l'andamento dell'esecuzione.

Compilazione e grado di finitura del software

Le funzioni fondamentali devono essere documentate con chiarezza (quelle *non* commentate dovrebbero essere auto-esplicative): occorre specificare che input prendono, qual è la loro funzione, se ci sono delle precondizioni, forme di dipendenza, di che tipo, etc. .

Il codice deve essere suddiviso in file sulla base di una articolazione funzionale. È necessario costruire un *Makefile* per la compilazione dei file costituenti il progetto; i file devono essere compilati con le opzioni *-Wall -pedantic* ; deve essere possibile compilare i moduli principali insieme (opzione di default) o separatamente.

L'elaborato finale deve contenere anche una *documentazione* in cui si illustra come eseguire i programmi (argomenti, dipendenze, etc.) che collettivamente costituiscono il sistema per la gestione delle prenotazioni e visite oggetto della presente esercitazione.

NB: il progetto non è da considerarsi definitivo fino a quando contiene la dicitura DRAFT.