SISTEMI OPERATIVI con LABORATORIO

Progetto della parte di Laboratorio – a.a. 2021-22

(le indicazioni per la consegna sono alla fine del documento)

Realizzare in linguaggio *C* ed in ambiente *Unix* (*Linux*, *macOS*, etc.) un'applicazione denominata ***AEROPORTO*** e strutturata nei seguenti processi:

- processo *Torre di Controllo* (di seguito ***Torre***)
- processo ***Hangar***
- ALMENO 10 processi ***Aereo***

Torre ed ***Hangar*** sono eseguiti all'avvio dell'applicazione, gli ***Aerei*** sono creati dal processo ***Hangar*** (*fork(2)*). L'applicazione deve poter essere interamente avviata da un solo terminale. Si precisa che il Docente valuterà l'applicazione eseguendola da un solo terminale a carattere, NON dall'ambiente grafico.

Descrizione generale dell'applicazione

L'AEROPORTO viene avviato eseguendo contemporaneamente ***Torre*** e ***Hangar***; quest'ultimo crea almeno 10 processi ***Aereo*** con *fork(2)*. ***L'AEROPORTO*** è dotato di ALMENO 2 piste di decollo. Ogni singolo ***Aereo*** richiede alla ***Torre*** l'autorizzazione al decollo e rimane in attesa passiva sin quando essa non la concede. La ***Torre*** concede le autorizzazioni al decollo in base all'ordine di arrivo delle richieste da parte degli ***Aerei*** (logica *FIFO*) ed in base alla disponibilità o meno di piste libere. Ogni singolo ***Aereo***, ricevuta l'autorizzazione dalla ***Torre***, decolla e, a decollo avvenuto, notifica alla ***Torre*** la fine del decollo (in modo che la Torre possa liberare la pista usata).

Dopo aver creato tutti gli ***Aerei*** il processo ***Hangar*** attende che essi terminino (*SC* elencate in *man 2 wait*), dopodiché notifica alla ***Torre*** che non ci sono più ***Aerei*** da far decollare e termina.

La ***Torre***, come già detto, attende le richieste di decollo e concede le relative autorizzazioni in base alla coda di ***Aerei*** che hanno fatto richiesta e alla disponibilità di piste libere. Una volta ricevuta da ***Hangar*** la notifica che non ci sono più ***Aerei*** che devono decollare la ***Torre*** termina e l'intera applicazione si chiude.

Ogni processo che fa parte dell'applicazione mostra a terminale dei messaggi che informano l'utente sui principali eventi che accadono durante l'esecuzione, come specificato di seguito per ogni processo. Ogni messaggio è preceduto dall'orario di accadimento ed è visualizzato nella forma:

HH:MM:SS nome processo: descrizione evento

(Aiuto: per ottenere l'ora corrente potete ispirarvi al seguente frammento di codice:

```
char s[256];
time_t timet;
time(&timet);
struct tm *pTm = localtime(&timet);
sprintf(s, "%02d:%02d:%02d", pTm->tm_hour, pTm->tm_min, pTm->tm_sec);
```

)

(ndr: modificato il 6 Maggio 2022)

Descrizione più in dettaglio dei processi componenti l'applicazione.

Processo ***Hangar***:
il processo *Hangar* viene avviato all'inizio dell'applicazione e genera con *fork(2)* almeno 10 processi figli che denominiamo ***Aerei***. Tra la creazione di un ***Aereo*** e la successiva devono trascorrere 2 secondi. Alla fine della generazione degli ***Aerei*** il processo ***Hangar*** deve attendere che tutti i processi ***Aereo*** generati siano terminati. Quando tutti i processi ***Aereo*** sono terminati ***l'Hangar*** notifica alla ***Torre*** che non ci sono più aerei che devono decollare e termina.
Eventi per i quali devono essere visualizzati messaggi a terminale:
- creazione del processo ***Hangar***
- creazione di ogni singolo ***Aereo***
- fine creazione ***Aerei***/inizio attesa terminazione degli ***Aerei***
- invio notifica "non ci sono più aerei da far decollare" alla ***Torre***
- terminazione del processo ***Hangar***

Processo ***Aereo***:
una volta creato, ogni ***Aereo*** attende un tempo variabile tra 3 e 8 secondi, determinato randomicamente, richiesto per la preparazione dell'aereo (possiamo interpretarlo come il tempo necessario per pulire l'aereo, fare i dovuti controlli dei dispositivi, imbarcare le vivande, etc.). Terminata la preparazione ogni ***Aereo*** richiede al processo ***Torre*** l'autorizzazione al decollo ed attende (per un tempo quindi indefinito) dalla ***Torre*** stessa il via libera prima di decollare.
Ottenuta l'autorizzazione decolla (in altre parole l'attesa indefinita s'interrompe). Il decollo dura un tempo variabile tra 5 e 15 secondi, determinato randomicamente. Trascorso il tempo necessario per il decollo il processo invia una notifica di fine decollo alla ***Torre*** e termina.
Eventi per i quali devono essere visualizzati messaggi a terminale:
- avvio del processo ***Aereo***
- inizio preparazione ***dell'Aereo*** con l'indicazione del tempo necessario
- avvenuto invio alla ***Torre*** della richiesta di autorizzazione al decollo
- inizio decollo
- fine decollo (terminazione)

(Aiuto: per ottenere un numero intero casuale compreso tra lMin ed lMax potete ispirarvi al seguente frammento di codice:

```
long get_random(long lMin, long lMax) {
      static int bSeed = 0;
      if(!bSeed) {
            struct timeval tv;
            gettimeofday(&tv, NULL);
            srandom(tv.tv_usec % 1000);
            bSeed = 1;
      }
      long l = random();
      return ((l % (lMax - lMin + 1L)) + lMin);

}
```

)

Processo ***Torre:***
il processo viene avviato all'inizio dell'applicazione ed attende le richieste di decollo da parte di ogni singolo

Aereo. ***L'AEROPORTO*** ha a disposizione ALMENO 2 piste di decollo. Quando la ***Torre*** riceve la richiesta di decollo da un ***Aereo*** inserisce tale richiesta nella lista delle richieste di decollo. Le richieste sono soddisfatte a seconda della posizione nella lista (la prima arrivata è la prima soddisfatta e così via) e a seconda della disponibilità o meno di una pista libera. Non appena c'è una pista libera la richiesta che da più tempo è nella lista (la prima arrivata di quelle rimaste) viene soddisfatta inviando ***all'Aereo*** corrispondente la notifica di autorizzazione al decollo. Ovviamente tale pista viene marcata come "occupata" (non si richiede che l'aereo sappia da quale pista sta decollando). Quando la ***Torre*** riceve la notifica di fine decollo da un ***Aereo*** marca come libera la pista da cui il decollo è avvenuto e, se presente, soddisfa un'altra richiesta di decollo nella lista. Quando riceve dal processo ***Hangar*** la notifica che non ci sono più ***Aerei*** anche la ***Torre*** termina.
Eventi per i quali devono essere visualizzati messaggi a terminale:
- avvio del processo
- ricevuta richiesta di decollo ***dall'Aereo*** x
- invio notifica di autorizzazione al decollo ***all'Aereo*** x dalla pista y
- ricevimento notifica di fine decollo ***dall'Aereo*** x
- ricevimento notifica che non ci sono più aerei che devono decollare da ***Hangar***
- terminazione processo

Per la comunicazione tra processi utilizzare una delle forme di *IPC* mostrate durante il Laboratorio (*pipe*, *socket*, *signals*, etc.).

Lo studente dovrà fornire anche uno script *Bash* che compila l'applicazione da zero e la avvia.

Tutta l'applicazione (processi e script) devono essere eseguiti dallo stesso terminale.

N.B.: quando si parla di "attese" s'intendono attese PASSIVE, cioè che non consumano tempo di *CPU* (NO cicli *while* per es.:), realizzate con *sleep(3)*, accessi bloccanti a risorse, o comunque con *system calls* e funzioni di libreria *C* che comportino la sospensione dell'esecuzione del processo. Inoltre, le attese per un tempo indefinito (es.: *Aereo* che attende la notifica di autorizzazione al decollo) non devono protrarsi oltre il necessario. *(ndr: modificato il 6 Maggio 2022)*

Sarà considerata nota di merito l'inserimento di commenti nel codice che ne facilitino la comprensione, così come l'inserimento di commenti atti a motivare le scelte di progetto dell'applicazione fatte.

Indicazioni per la consegna:
Il progetto deve essere spedito via email al Docente all'indirizzo fabio.rossi@unipg.it, oppure messo a disposizione su un qualsiasi *cloud* e reso scaricabile tramite *link* da inviare sempre al suddetto indirizzo.

La consegna deve avvenire tassativamente almeno una settimana prima della data dell'esame. Il formato può essere tar, tgz o zip.