Laboratorio di Sistemi Operativi
Progetto A.A. 2016-2017

Esercizio 1: scheduler di processi

In un sistema operativo, la gestione dei task (o processi) è una questione di primaria importanza che viene spesso riferita col nome di scheduling. Per i nostri scopi, un task è definito dalle seguenti informazioni: un identificatore numerico (ID), un nome (NOME TASK, stringa che può contenere al massimo 8 caratteri), una priorità (PRIORITA', numero compreso fra 0 e 9) e il numero di cicli di esecuzione che richiede per essere completato (ESECUZIONI RIMANENTI). Si supponga che un task possa richiedere al più 99 cicli di esecuzione per essere completato.

Lo scheduler è una funzionalità (del sistema operativo) che gestisce una coda in cui i task vengono mantenuti ordinati in base alla politica di scheduling adottata. In questo esercizio supporremo l'esistenza di due possibili politiche di scheduling:

- a) Scheduling su PRIORITA' (scheduling di default); si ordinano i task in modo decrescente rispetto al valore di PRIORITA' ad essi associato (quindi da 9 a 0). Il task da eseguire per primo sarà quello a cui è stato associato il numero di PRIORITA' più basso.
    - I task con uguale valore di PRIORITA' dovranno essere ordinati in modo decrescente rispetto al loro identificatore numerico ID.
- b) Scheduling su ESECUZIONI RIMANENTI; si ordinano i task in modo crescente rispetto al numero di ESECUZIONI RIMANENTI necessarie per il loro completamento. Il task da eseguire per primo sarà quello che necessita del maggior numero di ESECUZIONI RIMANENTI per essere completato.
    - I task con lo stesso numero di ESECUZIONI RIMANENTI dovranno essere ordinati in modo decrescente rispetto al loro identificatore numerico ID.

Scrivere e provare un programma in C che simuli la funzionalità di scheduling dei task di un sistema operativo. In particolare il programma, attraverso un menu di opzioni, dovrà consentire di:

1. inserire un nuovo task, richiedendo la sua priorità, il nome ed il numero di esecuzioni necessarie per il suo completamento (l'identificatore ID è univoco e viene assegnato automaticamente dal programma). Il task inserito dovrà poi essere posizionato nella coda a seconda della politica di scheduling utilizzata;
2. eseguire il task che si trova in testa alla coda;
3. eseguire il task il cui identificatore ID è specificato dall'utente;
4. eliminare il task il cui identificatore ID è specificato dall'utente;
5. modificare la PRIORITA' del task il cui identificatore ID è specificato dall'utente;
6. cambiare la politica di scheduling utilizzata, passando dalla a) alla b) e viceversa;
7. uscire dal programma.

L'esecuzione di un task con ESECUZIONI RIMANENTI = 1 comporta l'eliminazione del task stesso dalla coda, altrimenti il numero di ESECUZIONI RIMANENTI viene decrementato di 1 e l'ordinamento della coda aggiornato a seconda della politica di scheduling utilizzata.

Al termine di ogni operazione richiesta dall'utente, si deve stampare la coda risultante dei

task sotto forma di elenco ordinato, come nel seguente esempio (i task ad essere eseguiti per primi sono quelli verso il basso):

• caso di scheduling su PRIORITA'

ID	PRIORITA'	NOME TASK	ESECUZ. RIMANENTI
17	7	OUTPOST	3
2	5	SVCHOST	11
5	2	REGSVC	2
4	2	INTERNAT	3

(Il task con ID=4 sarà il primo ad essere eseguito)

• caso di scheduling su ESECUZIONI RIMANENTI

ID	PRIORITA'	NOME TASK	ESECUZ. RIMANENTI
5	2	REGSVC	2
17	7	OUTPOST	3
4	2	INTERNAT	3
2	5	SVCHOST	11

(Il task con ID=2 sarà il primo ad essere eseguito)

NOTA BENE:

- Il campo NOME TASK può contenere al massimo 8 caratteri.
- Il campo PRIORITA' è numero compreso fra 0 e 9.
- Il campo ESECUZIONI RIMANENTI è un numero compreso fra 1 e 99.
- Non essendo noto a priori il numero massimo di processi che il sistema operativo può eseguire contemporaneamente**, si richiede di utilizzare l'allocazione dinamica dei dati all'interno dello heap** (quindi utilizzando le istruzioni malloc, calloc, ...)

Esercizio 2: esecutore di comandi (versione sequenziale e parallela)

Scrivere un programma che continuamente legge dallo standard input righe di testo terminate da caratteri di fine riga (\n). Il programma termina quando inserisco una riga di testo vuota. Ciascuna riga rappresenta un comando esterno da eseguire, quindi contiene il nome del comando e gli eventuali argomenti:

```
comando argomento1 argomento2 ...
```

Il programma deve eseguire sequenzialmente i comandi (lanciati con gli eventuali argomenti indicati) in modo da salvare lo standard output di ciascun comando su di un file chiamato "out.k", ove k è il numero progressivo del comando eseguito.

Ad esempio, se il programma inizialmente legge dallo standard input le seguenti tre righe:
ls -la
date
cat /etc/passwd

allora il file "out.1" conterrà l'elenco dei file della directory, il file "out.2" conterrà la data e l'ora al momento dell'esecuzione, ed il file "out.3" sarà una copia del file "/etc/passwd".

Risolvere inoltre lo stesso esercizio supponendo però di voler eseguire i comandi non sequenzialmente ma in parallelo, ovvero generando n processi (dove n è il numero complessivo di righe di comando inserite) ognuno dei quali dedicato alla esecuzione di una riga di comando.

Si gestiscano anche le situazioni di errore che si possono verificare più frequentemente.

Esercizio 3: scambio di messaggi

Si realizzi una infrastruttura software per lo scambio di messaggi tra processi client (d'ora in avanti semplicemente “client”) gestita da un processo server (d’ora in avanti semplicemente “server”). La comunicazione tra client avviene nel seguente modo: un client può inviare un messaggio testuale a un altro client o ad un gruppo di client affidandolo al server, che si occupa di recapitarlo.

Tutte le comunicazioni fra tutti i processi in gioco andranno realizzate utilizzando segnali e pipe con nome.

Il server svolge le seguenti funzioni:

- mantiene la lista dei client connessi.
- riceve messaggi testuali da inoltrare ai client.

Il client esegue le seguenti attività:

- si connette al server, si disconnette dal server.
- invia e riceve messaggi testuali.

Il funzionamento del software dovrà essere il seguente: viene mandato in esecuzione il server, che rimane in attesa di informazioni da parte dei client (connessioni, disconnessioni, messaggi testuali da inoltrare). Vengono poi avviati alcuni processi client (ogni client sarà avviato in terminali distinti). Quindi i client propongono un menu con le scelte possibili, che sono:

1. Connessione, con la quale il client si registra presso il server.
2. Richiesta elenco ID dei client registrati, con la quale si richiede al server l'elenco dei client attualmente registrati.
3. Invio di un messaggio testuale a un altro client o a un insieme di client (specificandone l'ID).
4. Disconnessione, con la quale il client richiede la cancellazione della registrazione presso il server.
5. Uscita dal programma.

Il server riconosce il tipo di richiesta del client ed esegue ciò che essa prevede. In particolare, in corrispondenza di una richiesta di connessione il server assocerà al client un ID univoco di registrazione, che dovrà poi essere rimosso nel caso in cui lo stesso client richieda esplicitamente la disconnessione dal server (tramite la scelta 4) o nel caso in cui il client venga terminato (utilizzando la scelta 5 oppure premendo Ctrl-C da terminale).

Le infrastrutture da utilizzare per lo scambio di informazioni fra i processi sono le seguenti:

- un (unico) pipe con nome, creato dal server quando viene mandato in esecuzione, che viene utilizzato (in maniera condivisa) dai client in scrittura (per comunicare le proprie richieste al server) e dal server in lettura; il server è l’unico processo a cui è permessa la lettura dal pipe, i client potranno solo scrivere su tale pipe.
- un pipe con nome (uno per ogni client), creato appositamente dal server quando necessario, e utilizzato dal server in scrittura e dal client in lettura per inviare

l'elenco degli ID dei client attualmente registrati e per inviare messaggi testuali (il server usa il pipe associato al destinatario per recapitarlo).

- una serie di segnali, inviati dal server ai vari client, per notificare eventi quali la ricezione di un messaggio per un client (notifica al client destinatario) o l'errore per aver richiesto l'inoltro di un messaggio verso un client inesistente (notifica al client mittente).

Si gestiscano anche le situazioni di errore che si possono verificare più frequentemente.

Regole per la presentazione del progetto.

Il progetto assegnato è valido fino agli appelli di Febbraio 2018 compresi.
Il progetto deve essere svolto in gruppi di 2-3 persone. Gli studenti lavoratori possono svolgere il progetto individualmente.

E' necessario consegnare:

- Il **codice sviluppato** inclusi tutti i file necessari alla sua compilazione.
    - Il codice consegnato deve essere funzionante.
    - NOTA BENE: Affinché il progetto sia preso in considerazione per la valutazione, occorre che gli eseguibili possano essere compilati e mandati in esecuzione regolarmente sulle macchine dell'aula informatica 113 del Plesso Didattico Morgagni, in cui è installata su macchina virtuale Ubuntu 12.04.4 LTS a 32 bit, kernel 3.11.0.
- Una **relazione** sul progetto. Ogni gruppo di lavoro dovrà produrre **una sola relazione**, i cui autori saranno quindi tutti i membri del gruppo stesso. La relazione dovrà essere in formato **pdf**.
- Un archivio **.zip** oppure **.tar.gz** contenente il codice e la relazione dovrà essere caricato sul sito del corso seguendo l'apposito link che verrà reso disponibile alla pagina del corso.
- *Non si accettano consegne in formati differenti da quelli sopra indicati*

Relazione

La relazione dovrà contenere:

- Informazioni sugli autori (per ciascun componente del gruppo: Nome, Cognome, Numero di matricola, indirizzo e-mail) e la data di consegna.
- Descrizione dell'implementazione. In questa parte va inserito il **codice** implementato e commentato in modo chiaro ed esauriente. Sono consigliati commenti all'interno del codice, e commenti più ampi nella relazione. Alcuni possibili elementi specifici da discutere: descrizione dell'architettura software sviluppata e motivazione delle scelte progettuali, principali funzioni e strutture, principali algoritmi sviluppati.
- Evidenza del corretto funzionamento del programma rispetto alle specifiche fornite.
  In questa sezione saranno presentate e commentate alcune esecuzioni tipo del programma (es. in forma di screenshot), fornendo evidenza del suo corretto funzionamento per i casi ritenuti più critici o interessanti.
- **La relazione deve riportare tutte le istruzioni necessarie per compilare i file sorgenti e lanciare in esecuzione il programma**.

Frequently Asked Questions.

- Quando è la consegna del progetto? Quando si terrà l'orale? Dove?
    - *Luogo, data, orario saranno pubblicati e reperibili sul sito del corso.*
- Il progetto deve essere caricato da ciascun membro del gruppo o è sufficiente che venga caricato da un membro?
    - *Il progetto **deve** essere caricato da un solo membro del gruppo*
- Ho l'esame di XYZ nella stessa ora e data dell'orale di Laboratorio, cosa si può fare?
    - *Si farà tutto il possibile per evitare collisioni con altri esami del II anno. In caso di collisioni (esempio, con esami di altri anni) contattare il docente per definire una seconda data compatibilmente con le disponibilità del docente.*
- Ho consegnato adesso il progetto, ma posso fare l'esame in un altro appello?
    - *Sì certo. Il progetto, se ritenuto sufficiente, rimane valido fino agli appelli di Febbraio 2018 compresi.*
- L'intero gruppo deve sostenere l'orale nella stessa data, oppure è possibile effettuarlo in appelli diversi?
    - *E' possibile presentarsi in appelli differenti.*
- Non ho effettuato gli esami propedeutici del II anno, ma i miei colleghi di progetto sì. Cosa posso fare?
    - *La propedeuticità è necessaria. Deve attendere di effettuare gli esami propedeutici prima di poter effettuare l'esame di Laboratorio. I suoi colleghi di progetto possono presentarsi all'orale, ma Lei dovrà aspettare. Il progetto, se ritenuto sufficiente, rimane comunque valido fino agli appelli di Febbraio 2018 compresi.*
- È necessario prenotarsi per fare l'esame?
    - *No, la consegna vale automaticamente come prenotazione. **Si richiede di comunicare se intendete sostenere l'orale in un altro appello**.*