# Sistemi Operativi L-A
# Appello del 22 giugno 2009
# Turno 1
# Compito A

**1- Domande di Teoria**

a. Realizzazione della memoria virtuale.
b. Organizzazione di un sistema operativo: kernel monolitico e microkernel.

**2- Esercizio su System Call**

Si realizzi un programma di sistema, che, utilizzando le system call di Linux, preveda la seguente sintassi:

```
esame f  c1 c2 N
```

dove esame è il nome dell'eseguibile generato e:

- f è il nome assoluto di un file esistente e leggibile;
- c1 e c2 sono singoli caratteri;
- N è un intero positivo.

**Il processo iniziale P0** deve creare 2 processi figli P1 e P2; una volta creati i figli, P0 legge integralmente il file f, trasferendo ogni carattere letto a P1. Successivamente attenderà la terminazione dei figli per stamparne lo stato di terminazione.
**Processo P1:** filtra la sequenza di caratteri inviata dal padre allo scopo di eliminare le occorrenze di C1: ogni carattere diverso da C1 viene inviato da P1 a P2.
**Processo P2:** filtra la sequenza di caratteri proveniente da P1 allo scopo di eliminare le occorrenze di C2; ogni carattere diverso da C2 viene scritto sullo standard output. Dopo N secondi dalla sua creazione P2 dovrà comunque terminare forzatamente la sua esecuzione.

**3- Esercizio di Programmazione Concorrente in Java**

Un laboratorio clinico effettua esami radiografici (radiografie).
Il personale del laboratorio è costituito da:

- N tecnici che eseguono le radiografie;
- un medico radiologo, che esamina le radiografie e produce il referto;

Ogni tecnico, per ogni radiografia eseguita, inserisce le immagini in una busta che invia al radiologo tramite l'inserimento in una cassetta postale. La cassetta ha una capacità limitata a MAX buste.
Il medico radiologo per ogni busta estratta dalla cassetta, esamina le immagini e produce il referto, che viene inserito nella busta e depositato in un archivio (si supponga che la capacità dell'archivio non sia limitata).
Si realizzi un'applicazione concorrente nel linguaggio java in cui i tecnici e il medico siano rappresentati da thread concorrenti, nella quale la sincronizzazione sia realizzata tramite semafori.

# Sistemi Operativi L-A
## Appello del 22 giugno 2009
## Turno 1
## Compito B

**1- Domande di Teoria**

a. I thread: caratteristiche e realizzazione
b. Gestione dell'I/O: interazione tra processo e periferica nel caso di dispositivo abilitato alle interruzioni

**2- Esercizio su System Call**

Si realizzi un programma di sistema, che, utilizzando le system call di Linux, preveda la seguente sintassi:

```
esame f  c0 c1 N
```

dove esame è il nome dell'eseguibile generato e:

- f è il nome assoluto di un file esistente e leggibile;
- c0 e c1 sono singoli caratteri;
- N è un intero positivo.

**Il processo iniziale P0** deve creare il processo figlio P1, il quale a sua volta deve creare il processo nipote P2.

**Processo P2:** legge integralmente il file f, trasferendo ogni carattere letto a P1.
**Processo P1:** filtra la sequenza di caratteri inviata da P2 allo scopo di eliminare le occorrenze di C1: ogni carattere diverso da C1 viene inviato da P1 a P0. Dopo N secondi dalla sua creazione, P1 dovrà comunque terminare forzatamente la sua esecuzione.
**Processo P0:** dopo aver creato P1, P0 filtra la sequenza di caratteri inviata da P2 allo scopo di eliminare le occorrenze di C0: ogni carattere diverso da C0 viene scritto da P0 sullo standard output. Successivamente attenderà la terminazione del figlio per stamparne lo stato di terminazione.

**3- Esercizio di Programmazione Shell**

Si realizzi un file comandi Unix con la seguente interfaccia:

```
trova dir fileOut parola
```

dove:

- **`dir`** è un direttorio assoluto esistente nel filesystem,
- **`fileOut`** è un nome di file assoluto nel filesystem
- **`parola`** è una qualunque stringa.

Il file comandi si deve occupare di cercare, all'interno del solo direttorio **`dir`** (e cioè tralasciando tutti i sottodirettori di **`dir`**), tutte le occorrenze di **`parola`** all'interno di file regolari. Il file comandi deve scrivere le righe contenenti l'occorrenza di **`parola`** sul file **`fileOut`**.
Prima di terminare l'elaborazione, il file comandi deve stampare a video il numero di righe totali contenute su file **`fileOut`**.

Si rammenti l'utilizzo del comando grep per reperire stringhe all'interno di file.

# Sistemi Operativi L-A
# Appello del 22 giugno 2009
# Turno 2
# Compito A

**1- Domande di Teoria**

a. Scheduling della CPU in sistemi interattivi.
b. Allocazione della memoria nei sistemi multiprogrammati

**2- Esercizio su System Call**

Si realizzi un programma di sistema, che, utilizzando le system call di Linux, preveda la seguente sintassi:

```
esame M f  c0 c1
```

dove esame è il nome dell'eseguibile generato e:

- M è un intero positivo.
- f è il nome assoluto di un file esistente e leggibile;
- c0 e c1 sono singoli caratteri;

**Il processo iniziale P0** deve creare il processo figlio P1, il quale a sua volta deve creare il processo nipote P2.

**Processo P2:** legge integralmente il file f, trasferendo ogni carattere letto a P1. Dopo M secondi dalla sua creazione, P2 dovrà comunque terminare forzatamente la sua esecuzione.
**Processo P1:** filtra la sequenza di caratteri inviata da P2 allo scopo di eliminare le occorrenze di C1: ogni carattere diverso da C1 viene inviato da P1 a P0.
**Processo P0:** dopo aver creato P1, P0 filtra la sequenza di caratteri inviata da P2 allo scopo di eliminare le occorrenze di C0: ogni carattere diverso da C0 viene scritto da P0 sullo standard output. Successivamente attenderà la terminazione del figlio per stamparne lo stato di terminazione.

**3- Esercizio di Programmazione Shell**

Si realizzi un file comandi Unix con la seguente interfaccia:

```
contaRighe dir fileOut
```

dove **`dir`** è un direttorio assoluto esistente nel filesystem e **`fileOut`** un nome di file assoluto nel filesystem.

Il file comandi si deve occupare di contare il numero totale di linee di tutti i file regolari presenti all'interno del solo direttorio **`dir`** (e cioè tralasciando tutti i sottodirettori di **`dir`**).
Il file comandi deve poi scrivere il numero di linee totali sul file **`fileOut.`**

# Sistemi Operativi L-A
# Appello del 22 giugno 2009
# Turno 2
# Compito B

**1- Domande di Teoria**

a. Interazione tra processi nel modello ad ambiente locale
b. Caratteristiche e implementazione del file system nel sistema operativo Unix.

**2- Esercizio su System Call**

Si realizzi un programma di sistema, che, utilizzando le system call di Linux, preveda la seguente sintassi:

```
esame  M f  c1 c2
```

dove esame è il nome dell'eseguibile generato e:

- M è un intero positivo;
- f è il nome assoluto di un file esistente e leggibile;
- c1 e c2 sono singoli caratteri.

**Il processo iniziale P0** deve creare 2 processi figli P1 e P2; una volta creati i figli, P0 legge integralmente il file f, trasferendo ogni carattere letto a P1. Successivamente attenderà la terminazione dei figli per stamparne lo stato di terminazione.
**Processo P1:** filtra la sequenza di caratteri inviata dal padre allo scopo di eliminare le occorrenze di C1: ogni carattere diverso da C1 viene inviato da P1 a P2. Dopo M secondi dalla sua creazione, P1 dovrà comunque terminare forzatamente la sua esecuzione.
**Processo P2:** filtra la sequenza di caratteri proveniente da P1 allo scopo di eliminare le occorrenze di C2; ogni carattere diverso da C2 viene scritto sullo standard output.

**3- Esercizio di Programmazione Concorrente in Java**

Un piccola azienda artigianale produce orologi a elevata precisione.
Il personale dell'azienda è costituito da:

- N tecnici orologiai che costruiscono gli orologi;
- un certificatore, che verifica la precisione di ogni orologio prodotto e produce un certificato di qualità da allegare all'orologio.

Ogni tecnico invia ogni orologio prodotto al certificatore, depositandolo in un apposito carrello; il carrello ha una capacità limitata a MAX orologi.
Il certificatore, per ogni orologio estratto dal carrello, esegue su di esso i test di precisione e produce il certificato, che viene allegato all'orologio. Gli orologi completi di certificati vengono depositati in uno scaffale (si supponga che la capacità dello scaffale non sia limitata).
Si realizzi un'applicazione concorrente nel linguaggio java in cui i tecnici e il certificatore siano rappresentati da thread concorrenti, nella quale la sincronizzazione sia realizzata tramite semafori.