Sistemi Operativi L-A
Compito di Martedì 16 Dicembre 2008
CdS in Ingegneria Informatica - Prof. Paolo Bellavista

Compito A - Parte di Programmazione di Sistema (16 punti)

Si scriva un programma C che, utilizzando le System Call del sistema operativo UNIX, abbia un'interfaccia del tipo:

```
unicaPipeA searchCar fileIn
```

dove **searchCar** è un singolo carattere e **fileIn** un nome assoluto di file esistente nel file system. Dopo aver effettuato gli opportuni controlli sui parametri di invocazione, il processo iniziale **P0** deve generare due processi **P1** e **P2**, fratelli fra loro.

fileIn contiene una sequenza di caratteri organizzati in righe (numero di righe non noto a priori, lunghezza massima di una riga pari a 80 caratteri), con ciascuna riga che termina con il carattere '1' oppure con il carattere '2'. I processi figlio devono leggere concorrentemente l'intero **fileIn**, **P1** processando le sole righe che terminano con '1', **P2** occupandosi delle righe che terminano per '2'. In particolare, per ciascuna riga di pertinenza, **P1** e **P2** dovranno inviare un messaggio a **P0** contenenti le posizioni nella riga in cui il carattere **searchCar** è stato trovato (ad esempio, se **searchCar** è contenuto come primo, terzo e decimo carattere della riga, il messaggio dovrà contenere 0, 2 e 9). I processi figlio devono comunicare con il padre facendo uso di una unica pipe condivisa, utilizzata quindi sia da **P1** che da **P2** per scrivere messaggi a **P0** (due processi scrittori e uno lettore). **P0** deve semplicemente leggere i messaggi dalla pipe, contare quanti ne ha ricevuti da **P1** e quanti da **P2**, e scriverne il contenuto immutato sul file di nome relativo **tmp** in una directory qualunque del lab4 in cui si hanno diritti di scrittura.
Una volta terminata la ricezione di messaggi da parte di **P0**, **P0** deve comunicare sia a **P1** che a **P2** chi dei due processi ha inviato il maggior numero di messaggi (l'unica pipe condivisa è utilizzata questa volta in verso opposto) e terminare immediatamente il processo con conteggio più basso. L'altro processo dovrà invece scrivere su **tmp** "Sono il processo `pid` e ho inviato il numero maggiore di messaggi".
In ogni istante, deve essere inoltre possibile per l'utente forzare la modifica del comportamento del programma concorrente premendo la combinazione di tasti <CTRL-C>. In particolare, in seguito alla pressione di <CTRL-C> **P2** dovrà essere immediatamente terminato e **P1** dovrà occuparsi di tutte le righe rimanenti del file.

Si facciano le ipotesi semplificative desiderate in termini di modello affidabile dei segnali e di innestamento dell'esecuzione dei gestori associati. Come sempre, si facciano le scelte di sincronizzazione dei processi ritenute più opportune, cercando di sequenzializzare il meno possibile le varie operazioni richieste.

Sistemi Operativi L-A
Compito di Martedì 16 Dicembre 2008
CdS in Ingegneria Informatica - Prof. Paolo Bellavista

Compito A - Parte di Programmazione File Comandi (8 punti)

Si scriva un file comandi in Bash shell di Linux che abbia l'interfaccia:

`findDifferentUser <targetDir1>...<targetDirN>`

dove **`<targetDir1>...<targetDirN>`** sono nomi assoluti di directory esistenti nel filesystem. Si svolgano gli opportuni controlli sugli argomenti di invocazione del file comandi.

Il compito del file comandi è quello di ***esplorare le gerarchie individuate da ciascuna directory padre*** ***`<targetDirI>`***, ossia la directory i-esima stessa e i suoi sottoalberi.
Il file comandi deve analizzare il contenuto di ciascun direttorio esplorato alla ricerca di ***file di proprietà di un utente diverso dal proprietario della cartella padre*** della gerarchia corrente **`<targetDirI>`**. Al termine dell'esecuzione, per ciascuna cartella padre **`<targetDirI>`**, il programma dovrà stampare a video il ***numero delle occorrenze totali di file con proprietario differente*** da quello di **`<targetDirI>`** trovati nel sottodirettorio di pertinenza.

Ad esempio, supponendo di invocare il comando:

`findDifferentUser /home/user/firstDir`

e che il file system sia:

```
/home/user/firstDir (di proprietà di user)
/home/user/firstDir/file_di_user1.txt (di proprietà di user)
/home/user/firstDir/file_di_root1.sh (di proprietà di root)
/home/user/firstDir/a/
/home/user/firstDir/a/file_di_root2.txt (di proprietà di root)
/home/user/firstDir/a/
/home/user/firstDir/a/b/file_di_user2.txt (di proprietà di user)
/home/user/firstDir/a/b/file_di_root3.txt (di proprietà di root)
```

il programma fornirà come output:

```
/home/user/firstDir: trovati 3 file con owner diverso da user
```

NOTA: per determinare il proprietario di cartelle e file, si consiglia di utilizzare opportunamente il comando **`stat`**

Sistemi Operativi L-A
Compito di Martedì 16 Dicembre 2008
CdS in Ingegneria Informatica - Prof. Paolo Bellavista

Compito A – Domande di Teoria a Risposta Aperta Sintetica (8 punti)

1) Si illustri nel dettaglio che cosa si intenda per **modello affidabile dei segnali**. In particolare, si elenchino tutti gli aspetti che rendono l'implementazione Linux dei segnali non conforme al suddetto modello affidabile.

2) Si descriva qual è il ruolo della **tabella delle pagine** in un sistema operativo con memoria paginata. In particolare, si descriva la strutturazione e si illustrino i vantaggi/svantaggi delle soluzioni con cosiddetta tabella delle pagine "normale" vs. tabella delle pagine invertita.

3) Si mostri lo pseudo-codice che risolve il problema della sincronizzazione di task nell'accesso **produttore-consumatore a un buffer condiviso di dimensione N**, utilizzando opportunamente le operazioni **`wait()`** e **`signal()`** su semafori di Dijkstra. Si descriva inoltre l'esatto funzionamento dell'operazione **`wait()`** su di un semaforo.

Si rammenta ai candidati che durante la prova scritta non può essere consultato nessun tipo di materiale (slide del corso, appunti, libri, …); l'accesso al sito Web del corso è stato disabilitato.