Prima esercitazione di sistemi operativi
1997-1998
Memory manager di Minix

Nell'implementazione della system call EXEC di Minix, il MM controlla se e' gia` disponibile una zona libera di memoria sufficiente a contenere la nuova immagine di memoria, altrimenti la chiamata abortisce. Questa politica sara' chiamata "il vecchio algoritmo di EXEC".
Sarebbe invece meglio controllare se una zona libera sufficiente sara` disponibile una volta rilasciata la zona di memoria attualmente occupata dal processo che chiama l'EXEC: nel seguito lo chiameremo "nuovo algoritmo di EXEC".

Si chiede di inserire nel MM il nuovo algoritmo, ed inserire una nuova system call **,** la cui chiamata provochi il funzionamento di EXEC con il nuovo algoritmo (inizialmente, e finche' non avviene tale chiamata, i processi utilizzano sempre il vecchio algoritmo).
Sintassi: `int extend();`
il parametro di ritorno e` un codice di errore negativo (distinguendo le situazioni di errore se ne esistono), oppure 0 se OK.
Si osservi che se un processo chiama extend, dopo tale chiamata il nuovo algoritmo sara' applicato per tutti i processi ancora da attivare (anche dopo la terminazione del processo che ha chiamato extend).

Consegnare il seguente materiale:
- documentazione cartacea (vedi sotto);
- un dischetto di bootstrap del sistema Minix modificato;
- un secondo dischetto contenente, in due directories Minix distinte, il sorgente dei file modificati, sorgenti ed eseguibili dei programmi di test, piu' se necessari makefiles e scripts vari ecc.

Data di scadenza: lunedi' 19 gennaio 1998.
Inserire il materiale in una busta chiusa con sopra scritti numero del gruppo e nome-cognome-login di ogni componente e consegnarla **IN BUCA** a Dodero o Gianuzzi.
Tutto il materiale consegnato (documentazione e dischetti) dovra' essere **identificato** allo stesso modo (onde evitare disastri se qualcosa cadesse per terra, si rimescolasse il contenuto delle buste ecc).

Documentazione

1. Files sorgenti
Le modifiche apportate ai sorgenti Minix devono essere identificate da commenti "speciali" e commentate seguendo lo "stile" di Minix stesso.
Esempio /* modifica gruppo xx */ ... /* fine modifica gruppo xx */

2. Documentazione cartacea
La documentazione cartacea puo' essere consegnata manoscritta (purche' chiara e leggibile) oppure scritta con word processors, oppure meta' e meta', ecc.
Essa deve essere sintetica, in particolare non devono essere riportate informazioni contenute in questo foglio, sul Tanenbaum ecc.
La documentazione **deve contenere** una "copertina" che riporti nome, cognome, login,data e firma dei componenti il gruppo che hanno effettivamente svolto l'esercitazione.
Nel testo della documentazione devono essere presenti in modo chiaro le seguenti informazioni:

- informazioni di utente: deve essere possibile a chiunque sia in possesso dei dischetti e della documentazione, ricreare il kernel, ripassare i tests ecc.
- informazioni di sistema: chi fosse interessato a leggere i sorgenti C modificati, deve trovare informazioni su come e' stata fatta la modifica. Vanno elencati: i files contenuti nei dischetti, e per ciascun file le modifiche, sia inserimenti sia cancellazioni, apportate alle strutture dati e alle funzioni ivi contenute. Delle modifiche va data motivazione e descrizione sintetica. La descrizione va data in italiano o in pseudo-codice (bastano poche righe) Non va descritta l'operazione riga per riga! Verra' consegnato a chi ne fara' richiesta un esempio di cosa si intende per buona documentazione di una funzione Minix non compresa tra quelle facenti oggetto di questa esercitazione.

(*esempio di cosa NON scrivere*: "la funzione pinco() confronta il primo parametro con zero e se essi sono uguali, azzera anche il secondo parametro". Chiunque leggera' la documentazione sa anche leggere un IF!)

- informazioni sui tests: va motivata la scelta dei programmi di tests, e ne vanno riportate le modalita' di esecuzione. In particolare riportare i comandi per ricompilare e rieseguire i tests se sono semplici, altrimenti predisporre scripts che ricompilano, lanciano programmi in background ecc. Dei singoli programmi che compongono il test riportare solo le caratteristiche rilevanti (il sorgente dovrebbe essere comprensibile con l'aiuto al piu' di qualche commento).

(*esempio*: il programma P1 ha due loop annidati il cui scopo e' di ritardare il processo per almeno..., nel test n.2 si lanciano quattro copie di P1 e due di P2, e si osserva che ....)

- Va fornito infine un commento dei risultati ottenuti, ad esempio mostrando qualche situazione in cui il nuovo algoritmo riesce a lanciare piu' processi del vecchio algoritmo, oppure accade il viceversa, e perche' si verificano tali differenze.

Consigli utili (a chi non ha mai realizzato grossi programmi):

L'esercitazione si compone di tre parti:

- l'inserimento del nuovo algoritmo all'interno di MM, lasciando comunque il vecchio! Quindi occorre innanzi tutto comprendere bene il vecchio algoritmo;
- l'inserimento di una system call,
- la realizzazione di uno "stato" del MM che consente di passare al nuovo algoritmo dopo la system call.

Si raccomanda di progettare e realizzare separatamente le tre parti (non fare tutte le modifiche contemporaneamente!) Si raccomanda quindi di sviluppare versioni "bootabili" di Minix intermedie: ad es. una in cui e' definita la system call extend (ma non fa nulla), una in cui e' realizzato il nuovo algoritmo al posto del vecchio ecc.

Suggerimenti dettati dall'esperienza (e validi anche dopo la consegna):

1. Tenere sempre copie di tutto cio' che e' funzionante anche se incompleto (la catastrofe e' in agguato) e copie di modifiche parziali ecc. almeno come sorgenti.
2. Ripulire periodicamente le varie directories di tutte le prove non funzionanti ecc. (attenzione: potreste cancellare le versioni buone e lasciare quelle non funzionanti! Ma se non lo fate dopo un po' perdete il filo)
3. Fare degli script per tutte le sequenze di comandi non banali e ripetute sovente
4. Scrivere la documentazione man mano che si fanno le modifiche e tenerla "coerente" con le stesse.