Esercitazione finale per il corso
di Sistemi Operativi
(A.A. 2003/2004)

1

Descrizione dell'esercitazione

- L'esercitazione consiste nell'implementare un semplice sistema di "messaging", con cui gli utenti di una macchina possono inviare/ricevere messaggi ad/da altri utenti della macchina stessa.
- Nel seguito, chiameremo *messaged* il programma demone che fa funzionare il sistema, ricevendo e gestendo le richieste degli utenti.
- Il sistema puo' essere usato dagli utenti con i due comandi *invia* e *ricevi*

2

Il demone *messaged*

- Appena lanciato, il demone *messaged* alloca una coda di messaggi e un'area di memoria condivisa in grado di ospitare 10 messaggi di testo da 80 caratteri l'uno. (la memoria condivisa deve comunque avere dimensioni maggiori, per ospitare, insieme ad ogni messaggio, il mittente e il destinatario del messaggio, ed altre informazioni – si veda piu' avanti)
- *messaged* alloca anche i semafori necessari a far funzionare correttamente l'intero sistema.
- *messaged* comunica con i comandi degli utenti mediante la coda di messaggi (che d'ora in poi chiameremo *msgqueue*)

3

Comandi a disposizione degli utenti:

“invia”

- il comando *invia* viene usato così:
\$> invia login_name_destinatario messaggio
dove “messaggio” ha la generica forma:
parola1 parola2 parola3...
Per una lunghezza massima di 80 caratteri, inclusi gli spazi (messaggi piu’ lunghi verranno troncati). Ad esempio:
\$> invia st201020 ciao Marco, come stai?

4

il demone messaged

- Quando *messaged* riceve una richiesta da un comando *invia*, si forka, e mentre il padre torna a servire un’altra richiesta, il figlio controlla in memoria condivisa se c’è una entry libera per inserire il messaggio che arriva da *invia*.
- Se c’è spazio, il *messaged* figlio registra in memoria condivisa il messaggio insieme al nome del mittente e del destinatario. (ovviamente, questa operazione va eseguita in mutua esclusione)
- Infine invia su msgqueue all’istanza di *invia* che sta servendo un messaggio di risposta (vedi lucido successivo)

5

Comandi a disposizione degli utenti:

invia

- Il comando *invia* rimane in attesa, sulla msgqueue della risposta del *messaged* figlio, che può essere una delle seguenti due:
 - messaggio inviato
 - memoria piena, riprova più tardi
- *invia* stampa su video la risposta di *messaged* figlio e termina

6

Comandi a disposizione degli utenti (OPZIONALE)

- Comando **rimuovi** che, lanciato da *utente*, per ogni messaggio in memoria condivisa inviato in precedenza da *utente* lo visualizza e chiede a *utente* se vuole rimuoverlo dalla coda.

10

demone messaged: monitoraggio dei messaggi presenti

- Quando **messaged** riceve il segnale SIGUSR1 visualizza tutti i messaggi, con utente e destinatario (scegliete voi se visualizzare *solo* utente e destinatario, per questioni di privacy)...
- ... e (OPZIONALE) chiede all'utente se vuole cancellare tutti i messaggi in memoria (in realta' questo dovrebbe poter essere fatto solo da un utente privilegiato, ad esempio chi ha lanciato il demone, ma non complichiamoci troppo la vita. Se pero' qualcuno ci vuole provare...)

11

demone messaged e comandi utente: spegnimento del sistema

- Quando **messaged** riceve il segnale SIGUSR2 dealloca tutte le risorse prelevate e termina.
- Se i comandi **invia**, **ricevi** e **rimuovi** vengono lanciati quando il demone messaged non sta girando, devono rispondere che il sistema non e' attivato.

12

il demone *messaged*:
(OPZIONALE)

- Quando *messaged* riceve il segnale SIGUSR2 dealloca tutte le risorse prelevate e termina. Tuttavia, prima il demone salva in un file l'intero contenuto della memoria condivisa.
- Quando *messaged* viene fatto ripartire, legge il contenuto del file e lo porta in memoria condivisa. In questo modo, il sistema e' in grado di gestire i messaggi inviati e non ancora letti da una sessione di funzionamento alla successiva.

13

Monitoraggio del sistema:
OPZIONALE

- Il funzionamento del sistema dovrebbe essere monitorato su un opportuno file di log.
- Ogni richiesta dei comandi a disposizione degli utenti deve essere registrata sul file di log a cura del demone *messaged*.

14

Esempio di log file generato da
messaged

- data-ora: messaging system booting
- data-ora: prelievo risorse
- data-ora: SYSTEM READY!
- data-ora: st102030 invia messaggio a st203020
- data-ora: st234533 richiede lettura suoi messaggi
- data-ora: st234533 cancella messaggio
- data-ora: st234533 cancella messaggio
- ...
- ...
- data-ora: richiesto spegnimento del sistema.
- SHUTDOWN!

15

Osservazioni e consigli

- Probabilmente vi potrà essere utile dare un'occhiata alla soluzione del problema dei *processi lettori e scrittori* che abbiamo visto nella prima parte del corso.
- Gestite la memoria condivisa come un array bidimensionale, in cui cioè ogni entry corrisponde ad un array di caratteri. Ogni entry deve contenere un messaggio lungo al massimo 80 caratteri, più i nomi di mittente e destinatario, e il numero del messaggio (da 0 a 9).
- Per semplificare le cose, potete assumere che ogni campo delle 10 entry abbia lunghezza fissa. Messaggi più corti di 80 caratteri possono essere completati da caratteri ASCII "spazio".
- Potete usare un campo ulteriore per indicare se quella entry contiene un messaggio valido, o già cancellato.

16

Osservazioni e consigli

- Esempio di formato delle informazioni in memoria condivisa (come delimitare i primi 4 campi? basta uno spazio!):

0	v	st203040	st111111		ciao, come stai?
1	v	st903992	st199833		al pub alle 10 stasera?
2	i	st111111	st303022		niente cinema domani...
.....					
.....					
9	v	st203040	st688493		visto Valentino rossi!!!!?

17

Informazioni generali sull'esercitazione e sull'esame:

- Potete svolgere l'esercitazione in gruppo, di al massimo 4 persone (consiglio gruppi di 2 o 3 persone)
- *Scadenza per la consegna dell'esercitazione:* giugno 2005
- *Cosa si consegna:* listati del codice stampati, commentati in maniera adeguata. Sulla prima pagina, indicate chiaramente il nome, la matricola e l'e-mail di tutti gli studenti del gruppo
- *Quando:* quando mi fate vedere girare il programmi che avete scritto.

18

Informazioni generali
sull'esercitazione e sull'esame:

- Quando avete terminato l'esercitazione, fissate un appuntamento con me, via e-mail o passando nel mio ufficio, per quando mi farete vedere girare i programmi.
- Se avete sviluppato i programmi su Linux (o comunque non sulle macchine del laboratorio) e avete a disposizione un portatile, è possibile dimostrare il funzionamento dei programmi sul portatile.
- Se non avete un portatile, dovete portare il codice sulle macchine Solaris, e verificarne il corretto funzionamento (ricordatevi che ci sono leggere differenze tra diverse versioni di unix)

19

Informazioni generali
sull'esercitazione e sull'esame:

- Se il codice sviluppato gira correttamente secondo le specifiche che abbiamo visto, allora potete consegnarmi gli stampati del codice.
- Immediatamente, o fissando un secondo appuntamento (anche separatamente per ogni studente del gruppo), potete discutere l'esercitazione, in modo da dimostrare che avete partecipato al suo sviluppo, e dimostrare che avete una conoscenza di base dell'uso dei comandi Unix. (chi lo desidera puo' anche dare l'esame orale vero e proprio sull'intero contenuto del corso).

20

Informazioni generali
sull'esercitazione e sull'esame:

- UNICO VINCOLO:
 - la discussione dell'esercitazione puo' essere fatta SOLO dopo aver dato (e passato) lo scritto del corso
 - In questo modo conclusa la discussione posso assegnarvi il voto finale del corso di Sistemi Operativi e Laboratorio
- LE PARTI OPZIONALI:
 - se sviluppate correttamente possono contribuire ad assegnare un voto finale piu' alto.

21