Politecnico di Milano
FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE

Corso di Piattaforme Software per la rete – MODULO 2
anno accademico 2013-2014

Prof. William FORNACIARI

Cognome (leggibile).....	Nome (leggibile).....
Matr.....	Firma.....

D1	D2	D3	D4	TOT
8	9	8	7	32

SOLUZIONI APPELLO DEL 12 SETTEMBRE 2013

Quesito D1

1.1 In relazione alla shared memory lo studente descriva:

- Finalità e potenziali vantaggi/svantaggi rispetto ad altre forme di comunicazione
- Ciclo classico di utilizzo e ruolo delle principali primitive

Il minimo è: allocate, attach, detatch, deallocate

1.2 Si completi inoltre il seguente codice

```
#include <stdio.h>
#include <sys/shm.h>
#include <sys/stat.h>
int main ()
{
  int segment_id;
  char* shared_memory;
  struct shmid_ds shmbuffer;
  int segment_size;
  const int shared_segment_size = 0x6400;

/* Allocate a shared memory segment.  */
segment_id = shmget (IPC_PRIVATE, shared_segment_size,
                     IPC_CREAT | IPC_EXCL | S_IRUSR | S_IWUSR);
/* Attach the shared memory segment.  */
  shared_memory = (char*) shmat (segment_id, 0, 0);
  printf ("shared memory attached at address %p\n", shared_memory);
/* Determine the segment's size.  */
shmctl (segment_id, IPC_STAT, &shmbuffer);
  segment_size = shmbuffer.shm_segsz;
  printf ("segment size: %d\n", segment_size);
  /* Write a string to the shared memory segment.  */
  sprintf (shared_memory, "Hello, world.");
/* Deatch the shared memory segment.  */
  shmdt (shared_memory);
```

Quesito D2

Indicare quale sequenza di caratteri viene stampata a video dal seguente codice, giustificando la risposta attraverso l' analisi del funzionamento del programma.

```
#include <signal.h>
#include <setjmp.h>
#include <stdio.h>
jmp_buf land;

void fun(int param){
      longjmp(land,1);
}
int main(int argc, char* argv[]){
      int pass=0;
      char* dangling=NULL;
         struct sigaction ha,old_ha;
      ha.sa_handler=&fun;
         sigaction(SIGSEGV,&handler,&old_handler);
      goto label2;
label1: *dangling=0xFF;
         fprintf(stderr,"a");
      fprintf(stderr,"should not get here\n");
label2: setjmp(land);
         fprintf(stderr,"b");
      if (pass==0){
         fprintf(stderr,"c");
            pass=1;
            goto label1;
      }
      return 0;
         fprintf(stderr,"d");
}
```

In questo frammento di codice è presente un gestore per il segnale di SEGV ricevuto quando avviene un errore di segmento (segmentation fault), volto a ripristinare lo stato del programma in un punto stabile dell' esecuzione (cfr. sorgenti esaminati a lezione , disponibili su http://home.deib.polimi.it/barenghi/lib/exe/fetch.php?media=teaching:05_signals.tar.bz2).

Il flusso di esecuzione del codice è il seguente:
il programma inizializza un intero (pass) a zero e un puntatore (dangling) a NULL, a seguire aggancia la funzione fun come gestore del segnale SEGV.
Il programma salta quindi alla label 2 dove la setjmp salva lo stato corrente dell' esecuzione nel buffer land e di seguito stampa una 'b'.
Il controllo su pass ha successo e di conseguenza viene stampata una 'c', settato pass a 1 e il flusso passa alla label1.
Alla label 1 viene dereferenziato un puntatore a NULL con tentativo di scrittura, causando un segmentation fault, con la conseguente segnalazione da parte del sistema operativo al programma via SIGSEGV.
Il gestore del SIGSEGV viene invocato e la funzione longjump riporta lo stato di esecuzione del programma a dove la setjump l' aveva lasciato.
Viene quindi stampato 'b' dalla printf successiva e il controllo su pass questa volta non ha successo, dato che pass vale 1.
Il programma termina, senza stampare la 'd' post return che è a tutti gli effetti codice morto.

Quesito D3

Dato un host, avente una sola interfaccia di rete Ethernet (assumete il nome assegnato sia eth0), indicate la sequenza di comandi tramite la suite iproute2 per configurarla in modo che abbia indirizzo MAC 00:11:22:33:44:55, indirizzo IP 192.168.0.1 in una sottorete con maschera lunga 24 bit.
Assumete che la scheda di rete sia attiva al momento (stato UP dell interfaccia) e abbia un MAC diverso da quello sopra indicato.

Allo scopo di cambiare l' indirizzo MAC alla scheda è necessario spegnerla preventivamente via:

```
ip link set eth0 down
```

in seguito si cambia il MAC tramite:

```
ip link set address 00:11:22:33:44:55 dev eth0
```

e la si riaccende con:

```
ip link set eth0 down
```

Per prendersi cura dell' assegnazione dell' indirizzo/maschera di sottorete è quindi possibile procedere con

```
ip addr add 192.168.0.1/255.255.255.0 dev eth0
```

Quesito D4

Quale struttura di sincronizzazione scegliereste allo scopo di coordinare più attori nelle seguenti situazioni (giustificate la risposta brevemente, ma chiaramente):

- Una mini-bacheca dove un numero molto alto di lettori effettua letture frequenti, e un solo scrittore si occupa di effettuare gli aggiornamenti. Deve essere certo che i lettori riescano a leggere un messaggio entro un tempo finito.
- Una struttura dati su cui più scrittori vanno a compiere frequenti scritture, mentre un lettore singolo si occupa di processare i dati scritti rimuovendoli mano a mano che li legge. Eventuali perdite di dati per carenza di capienza della struttura non sono un problema critico, ma è fondamentale che gli scrittori non siano soggetti a deadlock.

Nel primo caso la richiesta che i lettori debbano leggere un messaggio in tempo finito impone che la struttura sia lock-free per i lettori, sincronizzata in modo sicuro. Considerato che c'è un solo scrittore, la struttura dati ideale è la Read-Copy-Update (RCU), che, in questo contesto rispetta le specifiche garantendo il minimo overhead.

Nel secondo caso, la struttura da utilizzare è un buffer circolare: in questo modo gli scrittori non corrono rischi di bloccarsi all' infinito a vicenda e la cancellazione dei dati vecchi viene effettuata automaticamente tramite riscrittura.