

## Lezione 17

*Modificare il programma del primo esercizio della Lezione 14, utilizzando i thread, in modo da implementare l'accesso esclusivo al file delle prenotazioni tramite regioni critiche gestite da mutex.*

Il programma riportato di seguito è stato ottenuto da quello della lezione 14, sostituendo le chiamate a `fcntl` con chiamate a `pthread_mutex_lock` e `pthread_mutex_unlock` (ovviamente invece di creare dei nuovi processi tramite la `fork`, si creano dei nuovi thread tramite la `pthread_create`):

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <pthread.h>

#define MAXLENGTH 3
char buf[MAXLENGTH];

/* Dichiarazione ed inizializzazione del mutex. */
pthread_mutex_t mutex=PTHREAD_MUTEX_INITIALIZER;

void prenota(int *n);
int acmebook();

main() {
    int n,n1,n2,fd;
    pthread_t thread1,thread2;

    while(1) {

        if((fd=open("prenotazioni.txt",O_RDONLY))===-1) {
            perror("Errore in apertura del file delle prenotazioni");
            exit(1);
        }

        /* Lettura dei posti disponibili. */
        if(read(fd,buf,MAXLENGTH)>0) {
            n=atoi(buf);

            if(n==0) {
                printf("Non ci sono piu' posti disponibili.\n");
                exit(0);
            }
        }
        else
            break;
    }
}
```

```

close(fd);
printf("Posti disponibili: %d\n",n);
/* Viene chiesto il numero di posti da prenotare dall'ufficio A. */
printf("Numero di posti da riservare dall'ufficio A: ");
scanf("%d",&n1);
/* Viene chiesto il numero di posti da prenotare dall'ufficio B. */
printf("Numero di posti da riservare dall'ufficio B: ");
scanf("%d",&n2);

/* Creazione dei thread per gli uffici A e B. */
if(pthread_create(&thread1,NULL,(void *)&prenota,(void *)&n1)!=0) {
    perror("Errore nella creazione del primo thread");
    exit(2);
}

if(pthread_create(&thread2,NULL,(void *)&prenota,(void *)&n2)!=0) {
    perror("Errore nella creazione del primo thread");
    exit(3);
}

/* Il padre attende la terminazione dei due thread. */
pthread_join(thread1,NULL);
pthread_join(thread2,NULL);
}

}

void prenota(int *n) {
    int i,error_code;

    for(i=0;i<*n;i++) {

        /* se acmebook restituisce -1 significa che non vi sono
        * posti disponibili per soddisfare la prenotazione.
        */
        error_code=acmebook();

        if(error_code==-1) {
            printf("Numero di posti insufficiente.\n");
            exit(error_code);
        }

    }

}

int acmebook() {
    int error_code=0,fd,n,i;

```

```

/* Apertura in lettura/scrittura del file delle prenotazioni */
if((fd=open("prenotazioni.txt",O_RDWR))==-1) {
    perror("Errore in apertura del file delle prenotazioni");
    exit(1);
}

/* Inizio della regione critica */
pthread_mutex_lock(&mutex);

if(read(fd,buf,MAXLENGTH)>0) {
    n=atoi(buf);

    if(n>0) {
        n--;
        sprintf(buf,"%d",n);

        /* Riposizionamento all'inizio del file prima della scrittura. */
        if(lseek(fd,0,SEEK_SET)==-1) {
            perror("Errore di riposizionamento nel file delle prenotazioni");
            exit(4);
        }

        /* Il buffer viene 'ripulito' da eventuali caratteri spuri. */
        for(i=strlen(buf);i<MAXLENGTH;i++)
            buf[i]=' ';

        /* Scrittura su disco del nuovo numero di posti disponibili. */
        if(write(fd,buf,MAXLENGTH)==-1) {
            perror("Errore in scrittura nel file delle prenotazioni");
            exit(5);
        }
    }
    else
        error_code=-1;
}

/* Fine della regione critica */
pthread_mutex_unlock(&mutex);

close(fd);
return error_code;
}

```

**N.B.:** per compilare correttamente il programma ricordarsi di includere l'opzione `-lpthread` nella linea di compilazione.