Lezione 11

- Implementare un meccanismo per decidere se una camera è vuota, modificando eventualmente la funzione `getoccupier` e il file `residents`. Scrivere una procedura `findfree` per trovare la prima camera libera.
- Scrivere le procedure
  - `freeroom` per rimuovere un occupante da una camera;
  - `addguest` per assegnare una camera ad un ospite, controllando che questa sia libera.
- Utilizzando le funzioni `getoccupier`, `freeroom`, `addguest`, and `findfree`, scrivere un programma `frontdesk` per gestire il file `residents`.

Nell'implementazione che segue si è deciso di identificare le camere vuote, memorizzando nel file residents una stringa composta da 40 spazi e dal carattere di newline finale. In tal modo confrontando quanto recuperato in lettura nel buffer `namebuf` da `getoccupier`, si può stabilire se una camera sia vuota o meno, a seconda se `namebuf` contenga una stringa composta da 40 spazi oppure no.

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

#define NAMELENGTH 41
#define NROOMS 10

char namebuf[NAMELENGTH];
int infile=-1;
/* Il vettore di caratteri blank_line conterra' la stringa
 * composta da 40 spazi piu' il terminatore '\0'.
 * Gli elementi vengono inizializzati dalla funzione initialize.
 */
char blank_line[NAMELENGTH];

char *getoccupier(int roomno)
{
  off_t offset;
  ssize_t nread;

  if(infile==-1 && (infile=open("residents", O_RDONLY))==-1)
    {
      return (NULL);
    }

  offset=(roomno-1)*NAMELENGTH;

  if(lseek(infile,offset,SEEK_SET)==-1)
    return (NULL);
```

```
  if ((nread=read(infile,namebuf,NAMELENGTH))<=0)
    return (NULL);

  /* crea una stringa rimpiazzando il newline con un terminatore null */
  namebuf[nread-1]='\0';
  return (namebuf);
}

/* La funzione initialize inizializza gli elementi di blank_line
 * e controlla la presenza del file residents nella cartella corrente:
 * Nel caso in cui non esista lo crea e lo inizializza con NROOMS righe
 * vuote (i.e., composte da 40 spazi e dal newline).
 */
void initialize()
{
  int i;

  for(i=0;i<NAMELENGTH-1;i++)
      blank_line[i]=' ';

  blank_line[i]='\n';

  if((infile=open("residents",O_WRONLY | O_CREAT | O_EXCL,0644))==-1)
    {
      blank_line[NAMELENGTH-1]='\0';
      return;
    }

  for(i=0;i<NROOMS;i++)

    if(write(infile,blank_line,NAMELENGTH)==-1)
      {
        printf("Error initializing the residents file\n");
        exit(1);
      }

  close(infile);
  infile=-1;
  blank_line[NAMELENGTH-1]='\0';
}

/* findfree controlla le righe del file residents
 * finche' non ne incontra una vuota o raggiunge la fine del file.
 * Nel primo caso restituisce il numero della stanza vuota al
 * chiamante, mentre nel secondo caso restituisce 0
 * ad indicare che non vi sono stanze libere.
 */
int findfree()
{
```

```
  int i;

  for(i=1;i<=NROOMS;i++)

      if(strcmp(getoccupier(i),blank_line)==0)
        {

          if(infile!=-1)
            close(infile);

          infile=-1;
          return i;
        }

  if(infile!=-1)
    close(infile);

  infile=-1;
  return 0;
}

/* writeline e' una funzione ausiliaria utilizzata sia da
 * freeroom che da addguest. In pratica questa funzione
 * scrive la stringa puntata dal parametro line nella riga
 * del file residents corrispondente alla camera avente come
 * numero il valore del parametro roomno.
 */
int writeline(int roomno,char *line)
{
  int offset;

  if((infile=open("residents",O_WRONLY))==-1)
      return -1;

  offset=(roomno-1)*NAMELENGTH;

  if(lseek(infile,offset,SEEK_SET)==-1)
    {
      close(infile);
      infile=-1;
      return -1;
    }

  line[NAMELENGTH-1]='\n';

  if(write(infile,line,NAMELENGTH)==-1)
    {
      line[NAMELENGTH-1]='\0';
      close(infile);
      infile=-1;
```

```
      return -1;
    }

  line[NAMELENGTH-1]='\0';
  close(infile);
  infile=-1;
  return 0;
}

int freeroom(int roomno)
{
  return writeline(roomno,blank_line);
}

int getline(char s[], int lim) {
  int i=0;
  char c;

  while((c=getchar())!='\n' && c!=EOF && i<lim-1)
    s[i++]=c;

  s[i]='\0';
  return i;
}

int addguest(int roomno)
{
  int i,status;

  /* Come prima cosa si controlla che la stanza avente
   * numero roomno non sia gia' occupata.
   */
  if(strcmp(getoccupier(roomno),blank_line)!=0)
    {
      printf("Room n. %d is not free\n",roomno);

      if(infile!=-1)
        close(infile);

      infile=-1;
      return 1;
    }

  if(infile!=-1)
    close(infile);

  infile=-1;

  /* Se la stanza avente numero roomno e' libera,
   * si invita l'utente ad inserire il nome dell'ospite.
```

```
   */
  printf("Guest name (max 40 characters): ");

  while(getline(namebuf,NAMELENGTH-1)==0)
    printf("Guest name (max 40 characters): ");

  for(i=strlen(namebuf);i<NAMELENGTH-1;i++)
    namebuf[i]=' ';

  namebuf[i]='\0';

  /* Si scrive il nome dell'ospite nella riga
   * della camera con numero roomno.
   */
  status=writeline(roomno,namebuf);

  if(!status)
    printf("Room n. %d has been assigned\n",roomno);

  return status;
}

/* getroom chiede all'utente di digitare il numero
 * di una camera (funzione ausiliaria).
 */
int getroom(const char *label)
{
  int roomno=-1;

  while(!(roomno>0 && roomno<11))
    {
      printf(label);
      getline(namebuf,NAMELENGTH);
      roomno=atoi(namebuf);
    }

  return roomno;
}

main()
{
  int choice=-1,i,roomno;

  initialize();

  while(1)
    {
      /* Menu del frontdesk. */
      printf("\nReception:\n");
      printf("1 - Guest list\n");
```

```
printf("2 - Find the first free room\n");
printf("3 - Free a room\n");
printf("4 - Add a new guest\n");
printf("5 - Quit\n\n");
printf("Make your choice: ");

/* Input della scelta dell'utente. */
choice=getchar();

/* Eliminazione dallo stream di input di eventuali
 * caratteri spuri digitati per errore dall'utente.
 */
while(getchar()!='\n');

/* A seconda della scelta dell'utente viene
 * effettuata l'operazione corrispondente.
 */
switch(choice) {
case '1':
  printf("\nGuest list:\n");

  for(i=1;i<=NROOMS;i++)
    {

      if(strcmp(getoccupier(i),blank_line)==0)
        printf("Room n. %2d: free\n",i);
      else
        printf("Room n. %2d: %s\n",i,namebuf);
    }

    if(infile!=-1)
      close(infile);

    infile=-1;
    break;
case '2':
  roomno=findfree();

  if(roomno==0)
    printf("\nThere are no free rooms\n");
  else
    printf("\nThe first free room is n. %d\n",roomno);

  break;
case '3':
  roomno=getroom("\nNumber of the room to free: ");

  if(freeroom(roomno)==-1)
    {
      printf("Error writing in the residents file\n");
```

```
            exit(1);
          }
        else
          printf("Room n. %d has been freed\n",roomno);

        break;
      case '4':
        roomno=getroom("\nNumber of the room to assign: ");

        if(addguest(roomno)==-1)
          {
            printf("Error writing in the residents file\n");
            exit(1);
          }

        break;
      case '5':
        exit(0);
      }

    }

}
```