Filesystem

Exam 2021/09/07 - Ex 4 (1.5 points)

Italiano

Si considerino i metodi di codifica delle informazioni su file. Si indichino quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Consider the methods for encoding information in a file. Indicates which ones among the following observations are correct (possibly more than one). Note that incorrect answers may imply a penalty on the final score.

Scegli una o più alternative: Choose one or more options:

1. ⚪ I caratteri in UNICODE sono sempre salvati su 32 bits / UNICODE characters are always stored on 32 bits.
2. ✅ UNICODE è un estensione di ASCII usata per rappresentare più di 256 caratteri / UNICODE is an extension of ASCII used to represent more than 256 characters.
3. ✅ I file ASCII sono spesso più compatti rispetto ai file codificati in UNICODE / ASCII files are often more compact than UNICODE files.
4. ✅ Ci sono molte versioni della tabella ASCII, per tenere traccia delle differenti lingue e notazioni / There are several versions of the ASCII table, to keep track of the different languages and notations.
5. ⚪ La codifica UNICODE è usata per salvare file in formato binario / The UNICODE encoding is used to store files in binary format.
6. ✅ I file binari sono spesso più compatti rispetto ai file codificati in UNICODE / Binary files are often more compact than UNICODE files.
7. ⚪ Le codifiche ASCII e UNICODE differiscono sempre / The ASCII and UNICODE encodings always differ

Exam 2021/01/29 - Ex 5 (2.0 points)

Italiano

Si supponga che il disco rigido di un piccolo sistema embedded sia costituito da 24 blocchi di 1 MByte, che tali blocchi siano numerati da 0 a 23, che il sistema operativo mantenga traccia dei blocchi liberi (occupati) indicandoli in un vettore con il valore 0 (1), e che la situazione attuale del disco sia rappresentata dal seguente vettore:

```
0 1 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 1 0 0
```

Si indichino quali delle seguenti affermazioni sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Suppose that the hard disk of a small embedded system is composed of 24 blocks of 1 MByte each, which are numbered from 0 to 23. Suppose that the operating system keeps track of the free (occupied) blocks indicating them in a vector with the value 0 (1), and that the current situation of the disk is represented by the following vector:

```
0 1 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 1 0 0
```

Indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

Scegli una o più alternative: Choose one or more options:

1. ✅ Un file di dimensione 3.5 MByte NON può essere allocato utilizzato una strategia di allocazione contigua A file with dimension 3.5 MByte CANNOT be allocated using a contiguous allocation strategy
2. ⚪ Un file di dimensione 3.5 MByte NON può essere allocato utilizzato una strategia di allocazione concatenata A file with dimension 3.5 MByte CANNOT be allocated using a linked allocation strategy

3. ✅ Con la strategia di allocazione contigua ottenuta mediante l'algoritmo BEST-FIT possono essere allocati nell'ordine i file F1 di 1.6 MByte, F2 di 1.9 MByte e F3 di 2.6 MByte With the contiguous allocation strategy based on the BEST-FIT algorithm, the following files can be allocated in the following order: F1 of 1.6 MByte, F2 of 1.9 MByte, and F3 of 2.6 MByte
4. ⬜ Con la strategia di allocazione contigua ottenuta mediante l'algoritmo FIRST-FIT possono essere allocati nell'ordine i file F1 di 1.6 MByte, F2 di 1.9 MByte e F3 di 2.6 MByte With the contiguous allocation strategy based on the FIRST-FIT algorithm, the following files can be allocated in the following order: F1 of 1.6 MByte, F2 of 1.9 MByte, and F3 of 2.6 MByte

Exam 2021/06/18 - Ex 4 (1.5 points)

Italiano

Si supponga che dei file siano allocati su un hard disk usando l'allocazione di tipo contiguo.
Si indichi quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale

English

Suppose files are allocated on a hard disk using contiguous allocation.
Please indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score

Scegli una o più alternative: Choose one or more options:

1. ✅ Essa soffre di frammentazione interna. It suffers from internal fragmentation.
2. ⬜ Ogni blocco contiene un puntatore al blocco successivo. Each block contains a pointer to the next block.
3. ✅ Essa soffre di frammentazione esterna. It suffers from external fragmentation.
4. ✅ L'accesso sequenziale è immediato, facile da realizzare. Sequential access is immediate, easy to achieve.
5. ⬜ I file possono crescere di dimensione fintantoché c'è spazio sull'hard disk. Files can grow in size as long as there is space on the hard disk

Exam 2022/09/06 - Ex 1 (2.0 points)

Italiano

Relativamente alle system call **open()**, **read()**, **write()** e **close()**, si indichino quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Regarding the system calls **open()**, **read()**, **write()** and **close()**, indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

Scegli una o più alternative: Choose one or more options:

1. ✅ La system call **write()** può essere utilizzata per scrivere byte indifferentemente su file, su schermo o su pipe. The **write()** system call can be used to write bytes either on a file, console or pipe without distinction.
2. ⬜ La system call **open()** accetta esattamente due parametri. The **open()** system call takes exactly two parameters.
3. ✅ Le system call **read()** e **write()** possono essere utilizzate rispettivamente per leggere e scrivere testo (caratteri) o sequenze di byte. The **read()** and **write()** system calls can be used respectively to read or write text (characters) or byte sequences.
4. ⬜ La system call **open()** accetta esattamente tre parametri. The **open()** system call takes exactly three parameters.
5. ⬜ La system call **read()** ritorna il numero di elementi letti invece del numero di bytes letti, come nel caso della funzione **fread()**. The **read()** system call returns the number of elements read instead of the number of bytes read, as it is done by the function **fread()**.

6. ✅ Nel caso di creazione di un file, attraverso la system call **open()** si possono impostare i permessi del file creato. When creating a file, the permissions of the newly created file can be set through the **open()** system call.

Exam 2022/06/14 - Ex 4 (2.0 points)

Italiano

Relativamente alle system call **stat()**, **lstat()** e **fstat()**, si indichino quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Regarding the system calls **stat()**, **lstat()**, and **fstat()**, indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

Scegli una o più alternative: Choose one or more options:

1. ⬜ Nel caso in cui ci sia un link simbolico `link_f1` che punta al file `f1`, la system call lstat() ritorna informazioni relativamente al file `f1`. If there is a symbolic link `link_f1` pointing to file `f1`, the lstat() system call returns information about file `f1`.
2. ✅ Le system call stat(), lstat() e fstat() sono tutte caratterizzate dagli stessi valori di ritorno. The system calls stat(), lstat(), and fstat() are all characterized by the same return values.
3. ✅ Tra le informazioni che possono essere ottenute dalle system call stat(), lstat() e fstat() vi è il numero di i-node relativo al file su cui la system call è stata eseguita. Among the pieces of information that can be obtained from the system calls stat(), lstat() and fstat() there is the i-node number associated with the file on which the system call has been executed.
4. ✅ Le system call stat() e fstat(), se applicate su un file di tipo regolare (cioè non un link simbolico), sono equivalenti. The system calls stat() and fstat(), if applied on a regular file (i.e., not a symbolic link), are equivalent.
5. ⬜ Le tre system call permettono di elencare le entries contenute in una directory. The three system calls allow us to list the entries contained in a directory.
6. ✅ Le tre system call restituiscono un puntatore ad una struttura di tipo `struct stat`. The three system calls return a pointer to a structure of type `struct stat`.

Exam 2022/06/14 - Ex 1 (5.0 points)

Italiano

Un file di tipo ASCII immagazzina una sequenza di record. Ogni record include 3 campi: un valore intero, una stringa e un numero reale. I tre campi hanno tutti una dimensione variabile e sono separati da un numero variabile di spazi. Il seguente è un esempio di file con tale formato:

```
15345 Acceptable 26.50
146467 Average 23.75
. . .
```

Si scriva un tratto di codice C che, utilizzando le system calls UNIX **open(), read()**, e **close()** per effettuare l'I/O da file, memorizzi tali record in un array di strutture di tipo `record_t`, definito nel modo seguente:

```
#define N 100

typedef struct record_s {
  int i;
  char s[N];
  float f;
} record_t;
```

Suggerimento: Si ricordi che la system call **read()** manipola i file in formato binario.

English

An ASCII file stores a sequence of records. Each record includes 3 fields: an integer value, a string, and a real number. All three fields have variable size and are separated by a variable number of white spaces. The following is an example of such a file:

```
15345 Acceptable 26.50
146467 Average 23.75
. . .
```

Write a segment of C code that, using the UNIX system calls **open(), read()**, and **close()** to perform file I/O, stores these records into an array of structures of type `record_t`, defined as follows:

```
#define N 100

typedef struct record_s {
  int i;
  char s[N];
  float f;
} record_t;
```

Suggestion: Keep in mind that the system call read() manipulates files in binary form.

Risposta: Answer:

```
typedef struct record_s {
  int i;
  char s[N];
  float f;
  struct record_s *next;  // Added
} record_t;

char c;
int fd, not_end1;
char line[MAXIMUM_LINE_LENGTH];
int position = 0;
record_t *r, *head;

// iterate till the end of file
head = NULL;
do {
  position = 0;
  do {
    not_end = read (fd, &c, sizeof(char));
    if (not_end != 0) {
      line[position] = c;
      position++;
    }
    // I suppose that newline are represented only by a '/n'
  } while (c!='\n' && not_end);

  r = (record_t *) malloc (1 * sizeof (record_t));
  sscanf (line, "%d %s %f", &r->i, r->s, &r->f);
  r->next = head;
  head = r;
} while (not_end);
```