Sistemi di Calcolo (A.A. 2022-2023)
Corso di Laurea in Ingegneria Informatica e Automatica
Sapienza Università di Roma

A

Compito (12/07/2023) – Durata 1h 30'

Inserire nome, cognome e matricola nel file `studente.txt`.

ISTRUZIONI PER STUDENTI DSA: svolgere a scelta due parti su tre.

Parte 1 (programmazione IA32)

Adler-32 è un noto algoritmo di checksum che viene utilizzato da diversi formati e librerie (come ZLIB). In questo esercizio, viene richiesto di tradurre in assembly un'implementazione semplificata del Adler-32. Nella directory E1, si traduca in assembly IA32 la seguente funzione C scrivendo un modulo `e1A.s`:

```
#include "e1A.h"

unsigned adler32(unsigned char *data, unsigned len) {
    if (data == NULL || len == 0)
        return 0;

    unsigned a = 1, b = 0;
    unsigned k;
    get_adler_constant(&k);

    int i;
    for (i = 0; i < len; ++i) {
        a = (a + data[i]) % k;
        b = (b + a) % k;    // Usare istruzione DIV
                            // e fare attenzione ai
                            // suoi vincoli sui registri!
    }
    return (b << 16) | a;
}
```

L'unico criterio di valutazione è la correttezza. Generare un file eseguibile `e1A` con `gcc -m32 -g`. Per i test, compilare il programma insieme al programma di prova `e1A_main.c` e `extra.s` fornito.

Nota: **non** modificare in alcun modo `e1A_main.c` e `extra.s`. Prima di tradurre il programma in IA32 si suggerisce di scrivere nel file `e1A_eq.c` una versione C equivalente più vicina all'assembly.

Parte 2 (programmazione di sistema POSIX)

Si vuole scrivere nel file `e2A.c` una funzione `vowelcount` con il seguente prototipo:

```
int vowelcount(const char** s, int n)
```

che, dato un array `s` contenente `n` stringhe, verifica il numero totale di vocali presenti in tutte le stringhe dell'array. Ad esempio, se l'array `s` è il seguente:

```
const char* s[] = {
    "I'm tired of weakness",
    "tired of my feet of clay",
```

```
    "tired of days to come",
    "tired of yesterday"
};
```

La funzione deve restituire il valore `27`, poiché l'intero array contiene complessivamente 27 vocali.

ATTENZIONE: l'implementazione della funzione **deve** seguire il seguente algoritmo:

1. Se `n` è uguale a 0 oppure `s` è `NULL`, la funzione deve restituire immediatamente il valore `-1`.
2. Altrimenti, la funzione crea `n` processi figli, dove il processo i-esimo conta il numero di vocali nella i-esima stringa di s; quando un processo figlio termina restituisce come codice di terminazione il numero di vocali conteggiate nella stringa analizzata. Le vocali da considerare sono (sia minuscole che maiusole): 'a', 'e', 'i', 'o', 'u'.
3. Il processo genitore attende la terminazione di tutti i processi figli uno alla volta e accumula il numero di vocali restituite da ciascun processo figlio.
4. Infine, il processo genitore restituisce il numero totale di vocali conteggiate in tutte le stringhe.

Per i test, compilare il programma insieme al programma di prova `e2A_main.c` fornito, che non deve essere modificato.

Parte 3 (quiz)

Si risponda ai seguenti quiz, inserendo le risposte (A, B, C, D o E per ogni domanda) nel file `e3A.txt`. Una sola risposta è quella giusta. Rispondere E equivale a non rispondere (0 punti).

Domanda 1 (cache)

Si consideri un sistema con una piccola cache completamente associativa contenente 2 sole linee da 16 byte ciascuna e politica di rimpiazzo LRU. Si assuma che l'array `v` sia allineato a un indirizzo multiplo di 16 byte e che la cache inizialmente non contenga alcun blocco di memoria in uso al processo. Quanti cache miss vengono generati dal seguente frammento di programma?

```
int v[12];
v[4] = 7;
v[8] = 2;
v[1] = 5;
v[11] = 9;
v[2] = v[8];
v[10] = v[1];
```

A	3	**B**	1
C	0	**D**	4

Motivare la risposta nel file `M1.txt`. **Risposte non motivate saranno considerate nulle**.

Domanda 2 (paginazione)

Si consideri un sistema di calcolo con spazio logico dei processi a 33 bit. Quanto occupa ciascuna pagina se la dimensione della tabella delle pagine è 4 MB? Si assuma che le entry della tabella delle pagine siano grandi ciascuna 16 bit.

A	32 KB	**B**	16 KB
C	4 KB	**D**	8 KB

Motivare la risposta nel file `M2.txt`. **Risposte non motivate saranno considerate nulle**.

Domanda 3 (permessi)

Che permessi (in notazione ottale) dovrebbe avere un file per essere accessibile in lettura e scrittura dall'utente proprietario, in lettura ed esecuzione dal gruppo proprietario, e solo in scrittura per tutti gli altri utenti?

A	`0632`	**B**	`0652`
C	`0754`	**D**	`0653`

Motivare la risposta nel file `M3.txt`. **Risposte non motivate saranno considerate nulle**.

Domanda 4 (Allocazione di memoria)

Si consideri un semplice allocatore di memoria dove, potendo scegliere, la `malloc` riusa il blocco libero con l'indirizzo più basso. Assumere per semplicità che non vi sia alcuna header e che i blocchi allocati vengono arrotondati a una dimensione multiplo di 4 byte. Inoltre, l'allocatore non divide blocchi liberi in più blocchi di dimensione inferiore, né fonde eventuali blocchi liberi adiacenti in un blocco di dimensioni superiori. Qual è il contenuto dell'heap alla fine della seguente sequenza di operazioni (`"X"` denota 4 byte allocati e `"."` denota 4 byte liberi)?:

```
p1 = malloc(6)
p2 = malloc(3)
p3 = malloc(14)
free(p2)
p4 = malloc(9)
free(p3)
p6 = malloc(20)
```

A	`XX|XXXXX|XXX`	**B**	`XX|.|....|XXX|XXXXX`
C	`XX|.|XXXX|...|XXXXX`	**D**	Nessuna delle precedenti

Motivare la risposta nel file `M4.txt`. **Risposte non motivate saranno considerate nulle**.