

Sistemi di Calcolo (A.A. 2023-2024)

Corso di Laurea in Ingegneria Informatica e Automatica
Sapienza Università di Roma

B

Compito (13/06/2024) – 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)

Il leet (o anche l33t, 31337 o 1337) è una forma codificata di inglese caratterizzata dall'uso di caratteri non alfabetici al posto delle normali lettere. In questo esercizio, si chiede di tradurre in assembly una funzione che decodifica stringhe in leet. Nella directory E1, si traduca in assembly IA32 la seguente funzione C scrivendo un modulo `e1B.s`:

```
void deleetify(unsigned char* v, unsigned len, unsigned char* res)
{
    unsigned i;
    unsigned char chr;

    for (i = 0; i < len; i++) {
        if (v[i] > 57) {
            res[i] = v[i];
        }
        else {
            deleetify_helper(v[i], &chr);
            res[i] = chr;
        }
    }
}
```

L'unico criterio di valutazione è la correttezza. Generare un file eseguibile `e1B` con `gcc -m32 -g`. Per i test, compilare il programma insieme al programma di prova `e1B_main.c`.

Nota: non modificare in alcun modo `e1B_main.c`. Prima di tradurre il programma in IA32 si suggerisce di scrivere nel file `e1B_eq.c` una versione C equivalente più vicina all'assembly. La funzione `deleetify_helper` ritorna il valore `''` in caso di input errato; quindi, se nella stringa di output avete il suddetto carattere è per via di un parametro errato.

Parte 2 (programmazione di sistema POSIX)

Si scriva una funzione che aiuti l'utente nella scelta di una possibile destinazione per una villeggiatura. Nello specifico, si scriva nel file `E2/e2B.c` una funzione con il seguente prototipo:

```
void destinazioniAcquistabili(struct connessione * list, const char
* partenza, int budget, const char * filename)
```

che, data in input la lista `list` contenente elementi di tipo `connessione`, una stringa `partenza`, un intero `budget`, ed un nome di file `filename`, scrive in `filename` l'elenco di tutte le destinazioni raggiungibili da `partenza` con un costo inferiore o uguale a `budget`.

Le righe scritte in `filename` devono avere il seguente formato:

DEF-UVZ-130

Ogni riga rappresenta una connessione tra un punto di partenza, un punto di arrivo, ed il relativo costo. Le tre informazioni sono separate dal carattere '-'. I collegamenti scritti nel file devono apparire in ordine di costo crescente. Nell'esempio, partendo da DEF è possibile raggiungere UVZ con un costo di 130€.

`list` è costituita da elementi rappresentati dalla struttura `connessione` definita nel file `e2B.h`. L'ordine degli elementi in `list` deve essere lo stesso con cui le corrispondenti connessioni compaiono in `filename`.

Per i test, compilare il programma insieme al programma di prova `e2B_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 (assembly)

Si consideri la seguente funzione:

```
char* foo(int size){  
  
    if(size<0)  
        exit(EXIT_FAILURE);  
  
    char* ret = (char*) malloc(size);  
    return memset(ret, 0x0, size);  
}
```

Qual è la dimensione minima che dovrà avere lo stack frame della funzione `foo()`, tenendo anche conto del return address?

A	16 bytes	B	13 bytes
C	12 bytes	D	20 bytes

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

Domanda 2 (threads)

Si consideri il seguente codice:

```
int count=0, n=5;  
  
void *thread_work(void *arg){  
    printf ("%d ", ++count);  
    return NULL;  
}  
  
int main() {  
    pthread_t* threads = (pthread_t*)malloc(n * sizeof(pthread_t));  
  
    int i;  
    for (i = 0; i < n; i++) {  
        if (pthread_create(&threads[i], NULL, thread_work, NULL) != 0)  
            exit(EXIT_FAILURE);  
    }  
    free(threads);  
    printf("Bye!\n");  
    return 0;  
}
```

Cosa viene stampato quando si esegue il programma?

A	1 2 3 4 Bye!	B	Bye!
C	1 5 Bye!	D	Dipende dalla singola esecuzione

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

Domanda 3 (Prestazione del software)

Se una porzione di codice impiega il 40% totale del programma, di quanto devo ottimizzare la parte restante in modo da avere uno speed-up totale del 60%?

A	267%	B	67%
C	280%	D	80%

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

Domanda 4 (programmazione POSIX)

Si consideri il seguente comando per comprimere una cartella:

```
user@pc:~$ zip -r cognome.nome.zip cognome.nome/
```

Cosa stamperebbe la seguente riga di codice, se fosse presente nel main del programma zip?

```
printf("%d, %s", argc, argv[1]);
```

A	3, zip	B	3, cognome.nome/
C	4, -r	D	4, cognome.nome.zip

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