



UNIVERSITÀ
di **VERONA**

Dipartimento
di **INFORMATICA**

Relazione progetto ASM

Bottacini Luca (VR474339)

Lecini Rustem (VR471559)

Roin Giovanni (VR471473)

06/05/2022

Indice

Sistema di telemetria F1	3
Traccia	3
Variabili	3
Funzioni	4
Passaggio dei parametri	4
Funzioni di supporto	5
Scelte progettuali	10

Sistema di telemetria F1

Abbiamo sviluppato un programma in linguaggio Assembly x86 con sintassi AT&T, che simula il sistema di telemetria del videogame F1.

Traccia

Il sistema legge un file la cui prima riga è il nome di un pilota, mentre le successive sono precedute dall'istante di tempo e dall'identificatore del pilota a cui sono associate e contengono:

- giri al minuto del motore
- temperatura del motore
- velocità istantanea del veicolo.

Tutte queste informazioni sono nel formato:

`istante, identif., velocità, giri, temperatura`

Se il pilota viene riconosciuto, dopo aver letto, interpretato ed elaborato le informazioni, il sistema produce un file in cui tutte le informazioni riguardano esclusivamente lui, altrimenti scrive "Invalid" all'interno dello stesso e termina.

Il riconoscimento avviene cercando il nome all'interno di un contenitore, se viene trovato ne viene utilizzato l'identificatore per interpretare le righe sottostanti.

Tutte le righe prodotte seguono il formato:

`istante, liv. giri, liv. temperatura, liv. velocità`

Tranne l'ultima che contiene i valori massimi di queste statistiche e in più la media (intera) della velocità istantanea, più precisamente:

`max. giri, max. temperatura, max. velocità, med. velocità`

I livelli che possono assumere le varie informazioni sono riportati nella tabella:

Parametro	LOW	MEDIUM	HIGH
Giri motore	$x \leq 500$	$5000 < x \leq 10000$	$x > 110$
Temperatura	$x \leq 90$	$90 < x \leq 110$	$x > 110$
Velocità	$x \leq 110$	$100 < x \leq 250$	$x > 250$

Variabili

Per riuscire a realizzare il sistema abbiamo usufruito di quattro variabili:

1. un array contenente i nomi dei piloti
2. un array contenente le stringhe "LOW", "MEDIUM", "HIGH"
3. un array contenente le soglie tra i livelli
4. una stringa per scrivere "Invalid".

L'array contenente i nomi dei piloti viene utilizzato nella funzione `telemetry` per capire se la prima riga del file è un pilota valido, mentre quelli contenenti le stringhe e le soglie vengono utilizzate nella funzione `telemetry_line` per stampare la giusta stringa in base al valore della statistica.

Questi array non sono altro che targhette salvate staticamente nel codice che poi vengono caricate una dopo l'altra contigualmente nello stack. Ad esempio per caricare le stringhe dei livelli abbiamo scritto:

```
/* Carica la stringa "HIGH". */
leal level_02_str, %esi      # Carica l'indirizzo in %esi.
movl %esi, -4(%ebp)         # e lo copia nello stack.

/* Ripete la stessa operazione con "MEDIUM" e "LOW". */
leal level_01_str, %esi
movl %esi, -8(%ebp)
leal level_00_str, %esi
movl %esi, -12(%ebp)
```

Mentre per l'array delle soglie, essendo costituito da interi, non abbiamo utilizzato l'istruzione `leal` ma `movl`. Infine, la stringa "Invalid" viene usata anch'essa in `telemetry` nel caso in cui il pilota non sia stato riconosciuto.

Funzioni

Passaggio dei parametri

Prima di tutto, durante la chiamata di una funzione esistono due protagonisti, il **chiamante** ed il **chiamato**, che hanno responsabilità diverse.

Più precisamente, nello standard di chiamata C, il *chiamante* deve:

1. Eventualmente salvare i registri `%eax`, `%edx` ed `%ecx`
2. Caricare nello stack dall'ultimo al primo i vari parametri
3. Chiamare la funzione stessa
4. Recuperare il risultato dal registro `%eax` se necessario.

Viceversa, il *chiamato* ha il compito di:

1. Salvare il registro `%ebp` nello stack
2. Allocare eventuale spazio nello stack per le variabili
3. Salvare i registri `%ebx`, `%esi`, `%edi` se vengono utilizzati
4. Eseguire tutte le operazioni utili a produrre il risultato
5. Recuperare i registri se precedentemente salvati
6. Liberare lo spazio se precedentemente allocato
7. Recuperare il valore iniziale di `%ebp` dallo stack
8. Ritornare al chiamante.

Funzioni di supporto

Per riuscire a implementare il sistema nel modo più comprensibile e ordinato possibile, abbiamo implementato delle funzioni di supporto come ad esempio le `strncmp` e `strncpy` della libreria standard C con alcune modifiche.

Lunghezza di una stringa

```
unsigned int
strlen(const char *str)
{
    // Copia l'indirizzo originale.
    const char *s = str;

    // Finché non incontra il terminatore
    // incrementa il puntatore.
    while (*s != 0)
        ++s;

    // Restituisce la distanza tra i due.
    return s - str - 1;
}
```

Calcola la lunghezza di una stringa escludendo il carattere terminatore.

Confronto tra stringhe

```
int
strncmp(const char *str1, const char *str2, unsigned int num)
{
    char c1 = 0, c2 = 0;

    // Copia un carattere da ciascuna di ognuna
    // delle stringhe finché i due caratteri estratti
    // corrispondono, non è terminata almeno una delle
    // due stringhe oppure finché il contatore dei
    // confronti non è a zero.
    do {
        c1 = *str1++, c2 = *str2++;
    }
    while (num-- > 0 && c1 != 0 && c1 == c2);

    // Restituisce la differenza tra i due caratteri.
    return c1 - c2;
}
```

Confronta un certo numero di caratteri di due stringhe.

Ribaltamento di una stringa

```
char *
strnrev(char *str, int num)
{
    // Copia l'indirizzo del primo carattere
    // e l'indirizzo dopo num caratteri.
    char *b = str, *e = str + num, c = 0;

    // Finché non raggiunge la metà della stringa
    // scambia i due caratteri e sposta i puntatori
    // verso il centro.
    for (; b < e; ++b, --e) {
        c = *b;
        *b = *e;
        *e = c;
    }

    return str;
}
```

Scambia un certo numero di caratteri di una stringa.

Copia di una stringa

```
int
strncpy(char *dst, const char *src, int num)
{
    // Copia l'indirizzo di destinazione.
    char *d = dst;

    // Copia il carattere di src in d finché non raggiunge
    // il terminatore oppure il contatore è nullo.
    while (num-- > 0) {
        if ((*d++ = *src++) == 0)
            break;
    }

    // Restituisce il numero di caratteri copiati.
    return d - dst;
}
```

Copia un certo numero di caratteri da una stringa in un'altra.

Solamente se il numero di caratteri da copiare supera la lunghezza della stringa sorgente, allora viene copiato anche il terminatore.

Copia di una stringa

```

unsigned int
strncpy(char *dst, const char *src, unsigned int num);
{
    // Copia il numero di caratteri da copiare.
    int n = num;
    // Copia l'indirizzo di destinazione.
    char *d = dst;

    // Copia il carattere di src in d finché non raggiunge
    // il terminatore oppure il contatore è nullo ma copiando
    // un carattere in meno.
    while (--num > 0) {
        if ((*d++ = *src++) == 0)
            break;
    }

    // Se ha copiato effettivamente almeno un carattere,
    // termina la stringa.
    if (num == 0 && n > 0)
        *d = 0;

    // Restituisce il numero di caratteri copiati.
    return d - dst;
}

```

Copia un certo numero di caratteri da una stringa in un'altra.

Indipendentemente dallo stato della stringa sorgente, quella di destinazione viene terminata sempre, al prezzo di un carattere copiato in meno.

Conversione da stringa ad intero

```

unsigned int
strtoi(const char *str)
{
    int res = 0;

    // Finché il carattere è numerico lo converte,
    // shift il numero e inserisce la nuova cifra.
    while (*str >= '0' && *str <= '9')
        res = res * 10 + *str++ - '0';

    return res;
}

```

Converte una stringa in un intero di base 10.

Conversione da intero a stringa

```
char *
itostr(unsigned int num, char *str)
{
    // Copia l'indirizzo originale.
    char *s = str;

    // Se il numero da convertire è maggiore di zero, allora
    // converte la cifra meno significativa e la copia in fondo
    // finché il numero non rimane azzerato.
    if (num > 0) {
        do {
            *s++ = (num % 10) + 48;
        }
        while (num /= 10);
    }
    // Altrimenti copia semplicemente zero.
    else
        *s++ = 48;

    // Termina la stringa in ogni caso.
    *s = 0;

    // Rovescia la stringa.
    strnrev(str, s - str - 1);
    return str;
}
```

Converte un intero di base 10 in una stringa.

Collocazione di un valore in un intervallo

```
unsigned int
select(unsigned int val, unsigned int arr[], unsigned int len)
{
    for (int i = 0; i < len; ++i)
        // Se il valore è minore della i-esima soglia, allora
        // restituisce l'indice.
        if (val <= arr[i])
            return i;

    // Restituisce l'indice più grande, non è in alcun intervallo.
    return len;
}
```

Verifica in quale intervallo si trova un determinato valore.

Separazione di una stringa

```
char*
strsep(char **ptr, char sep)
{
    // Copia l'indirizzo originale.
    char *s = *ptr, *d = *ptr;

    if (s != 0) {
        // Scorre la stringa finché non termina o non
        // raggiunge il carattere separatore.
        while (*s != 0 && *s != sep)
            ++s;

        // Se la stringa continua, la termina e aggiorna il puntatore.
        if (*s != 0) {
            *s++ = 0;
            *ptr = s;
        }
        // Altrimenti annulla la stringa.
        else
            *ptr = 0;
    }

    return d;
}
```

Spezza una stringa sul posto in base ad un certo carattere separatore.

Separazione di una stringa

```
unsigned int
strnsep(char *arr[], unsigned int num, char **ptr, char sep)
{
    unsigned int cnt = 0;

    for (unsigned int i = 0; i < num; ++i) {
        // Separa la stringa e se riceve un puntatore valido,
        // incrementa il contatore delle separazioni efficaci.
        if ((*arr++ = c_strsep(ptr, sep)) != 0)
            ++cnt;
    }

    return cnt;
}
```

Spezza una stringa sul posto in base ad un certo carattere separatore più volte.

Ricerca di una stringa in un array

```
int  
arrfind(const char *arr[], unsigned int len, const char *key)  
{  
    // Calcola la lunghezza della chiave.  
    int num = strlen(key);  
  
    // Scorre tutto l'array confrontando ogni elemento  
    // con la chiave, se almeno un elemento corrisponde  
    // ne restituisce la posizione.  
    for (int i = 0; i < len; ++i) {  
        if (strncmp(key, arr[i], num) == 0)  
            return i;  
    }  
  
    // Se non ha trovato alcun elemento restituisce  
    // un indice inesistente.  
    return -1;  
}
```

Cerca una particolare chiave all'interno di un array.

Scelte progettuali