Università degli Studi di Verona

DIPARTIMENTO DI INFORMATICA
Corso di Laurea in Informatica

Architettura degli Elaboratori

Relazione Elaborato Assembly

Candidati:
Marco Strambini
Matricola VR363938

Andrea Olivieri
Matricola VR353886

Anno Accademico 2013-2014

Indice

1 Introduzione **5**

2 Funzionamento generale **7**
2.1 Sottomenù 8

3 Componenti Software **11**
3.1 main 11
3.2 user_mode e superuser_mode 12
3.3 menu 13
3.4 scelta e reset 13

4 Funzioni generiche **15**
4.1 Pseudo codice di tutto il programma 16

5 Scelte progettuali **19**

Capitolo 1

Introduzione

Il progetto è incentrato sulla realizzazione di un menù di controllo di un cruscotto per auto; questo deve permettere di eseguire delle operazioni da parte di due tipologie di utenti: un "user" e un "superuser".

L'utente di tipo *user* ha meno possibilità di controllo, in quanto non deve essere in grado di modificare impostazioni importanti del sistema; infatti il suo menù è composto da 6 voci.

L'utente di tipo *superuser* ha pieno controllo del sistema, quindi può eseguire operazioni importanti dal punto di vista del funzionamento dell'automobile.

Come strategia di progettazione, si è deciso di dividere il codice in file che eseguono delle operazioni specifiche; si è inoltre cercato di non ripetere operazioni ricorrenti, trasformando queste operazioni in funzioni ben definite. Inoltre si è cercato di ottimizzare l'esecuzione del programma cercando di utilizzare più registri possibile piuttosto dello stack e variabili (quindi accessi alla memoria) in modo da rendere più veloce l'esecuzione del programma.

Capitolo 2

Funzionamento generale

Il programma deve essere eseguito da riga di comando, in modo da ricevere dei dati al momento dell'esecuzione, non durante lo svolgimento. Una volta avviato, il programma può assumere due differenti comportamenti:

- se è stata inserita la password corretta (2244), il programma si attiva in modalità *superuser* e verrà presentato un menù con 8 voci;
- se da riga di comando è stato avviato semplicemente il programma senza parametri di ingresso, il programma si avvia in modalità *user*, presentando quindi un menù di 6 voci.

I diversi menù verranno presentati riga per riga; l'utente quindi ha bisogno di scorrere il menù verso l'alto, verso il basso e nei sottomenù. Per eseguire queste operazioni è necessario utilizzare i tasti **e** (su), **d** (giù), **r** (sottomenù) seguiti dal tasto **enter** in modo da far capire alla chiamata di sistema di lettura da tastiera che la stringa inserita è completa. I due differenti menù sono i seguenti:

per user:

```
Assembly] > ./run.x
Starting
USER MODE: ON
> d
     1. Setting automobile:
> d
     2. Data: 15/06/2014
> d
     3. Ora: 15:32
> d
     4. Blocco automatico porte: OFF
> d
     5. Back-home: OFF
> d
     6. Check olio
>
```

per superuser:

```
Assembly] > ./run.x 2244
Starting
SUPERVISOR MODE: ON
> d
     1. Setting automobile (supervisor):
> d
     2. Data: 15/06/2014
> d
     3. Ora: 15:32
> d
     4. Blocco automatico porte: OFF
> d
     5. Back-home: OFF
> d
     6. Check olio
> d
     7. Frecce direzione
> d
     8. Reset pressione gomme
>
```

2.1 Sottomenù

Per poter cambiare le impostazioni, come da specifiche è stato implementato un sottomenù per le voci (4), (5), (7), e (8) in modo da settare il blocco automatico delle porte, il back-home, il numero di lampeggii delle frecce di direzione e il reset della pressione gomme; per raggiungerlo è necessario premere il tasto '**r**' e '**invio**'. I rispettivi sottomenù sono i seguenti:

```
> d
     1. Setting automobile (supervisor):
> d
     2. Data: 15/06/2014
> d
     3. Ora: 15:32
> d
     4. Blocco automatico porte: OFF
> r
          Impostazione corrente: Blocco automatico porte: OFF
     >
> d
     5. Back-home: OFF
> r
          Impostazione corrente: Back-home: OFF
     >
> d
     6. Check olio
> d
     7. Frecce direzione
> r
          Impostazione corrente: Numero lampeggii: 3
          set Numero lampeggii modalita' autostrada: 3
        Numero lampeggii settata
> d
     8. Reset pressione gomme
> r
```

```
     > r
          Pressione gomme resettata
>
```

Se durante l'esecuzione non si premono i pulsanti '**e**','**d**' o '**r**' si hanno due casi:

- se si preme il tasto invio, il programma termina;
- se si inserisce una stringa qualsiasi (massimo 50 caratteri) il programma termina con un messaggio di errore.

Capitolo 3

Componenti Software

Il programma è organizzato in tanti file quante sono le voci del menù in cui si ha un differente comportamento (le voci con i sottomenù) e in funzioni che rappresentano il codice ricorrente nell'esecuzione del programma.

Ogni funzione si trova in un file differente; è stato utilizzato un Makefile per compilare il programma e tenere traccia delle dipendenze tra file; in questo modo alla minima modifica, basta eseguire il comando make e ricompilare quindi le parti modificate.

Il programma parte dal codice scritto nel file *main.s*, da cui vengono chiamate le rispettive funzioni di menù per l'utente user e per quello supervisor. Queste due funzioni sono molto simili nella loro esecuzione, poiché eseguono circa le stesse operazioni, si differenziano solamente per la presentazione del menù (la prima presenta 6 voci mentre l'altra 8).

3.1 main

Questa funzione, che si trova nel file *main.s*, contiene il codice start, quindi è l'effettivo punto di partenza dell'esecuzione del programma. Il suo scopo è quello di far partire il programma leggendo dallo stack la password (se è presente). Lo stack iniziale si presenta in questo modo:

...
password
numargs
run.x

quindi, per capire se è presente la password, basta guardare il numero di argomenti passati (numargs):

- se è 1 allora non è stata inserita la password, quindi verrà attivata la modalità user;
- se è 2 allora è stata inserita la password e deve esserne controllata la correttezza: se è corretta allora si attiverà la modalità superuser, altrimenti il programma terminerà con un errore.

Bisogna considerare numargs > 0 in quanto alla prima posizione dello stack si trova il nome del programma.

in pseudocodice:

```
check_stack();

if (stack.numargs < 2)
        activate_user_mode();
else
        if(stack.arg2 == password)
                activate_superuser_mode();
        else
                exit_with_error();
```

3.2 user_mode e superuser_mode

Queste funzioni, che si trovano rispettivamente nei file user_mode.s e superuser_mode.s, hanno lo stesso comportamento, differiscono solo per il valore di una variabile inizializzata (superuser).

Per far identificare questi due funzioni dalle funzioni chiamate, è necessario vedere il contenuto della variabile superuser (che verrà passata come argomento tramite registri); se questa variabile vale 0, allora si è in modalità user, altrimenti si è in modalità superuser.

All'inizio dell'esecuzione, viene stampata la stringa corrispondente alla modalità di attivazione del menù (stringa 0); ogni volta che si preme un tasto direzione, viene stampata la corrispettiva voce (se tasto viene premuto il tasto e si scorre il menù verso l'alto, d verso il basso, r si passa al sottomenù). Il controllo del carattere viene eseguito dalla funzione compare_key.

Entrambi i tipi di menù tengono memorizzati i valori modificabili durante l'esecuzione del programma dentro variabili; per passare il valore di queste variabili alle funzioni chiamate, vengono utilizzati i registri.

Bisogna sempre considerare il primo carattere della stringa ricevuta, in quanto se si inserisce ad esempio la lettera 'e' la stringa di lettura conterrà 'e\n'.

in pseoudocodice:

```
read_string();

if (string.length  == 2)

        switch(string[0]){

                case 'e':       scroll_up();
                case 'd':       scroll_down();
                case 'r':       goto_submenu();

                default:        error();
        }
```

Quando è stata data una direzione di scorrimento (tramite tasti direzione), si chiama la funzione delegata alla stampa del menù e gli si passano, tramite registri, i parametri memorizzati durante l'esecuzione del programma.

```
prepare_to_call_menu:
        # preparo i registri per passare
        # i parametri alla funzione menu

        movl %edx, (counter)

        movl %edx, %eax

        movl superuser, %ebx

        movl lock_door, %ecx

        movl back_home, %edx

        call menu
```

3.3 menu

Questa funzione, che si trova nel file *menu.s*, si occupa di stampare a video le differenti righe del menù. In base al carattere precedentemente letto, viene incrementato il contatore della riga da stampare (ad ogni riga è associato un numero) in modo che si tenga traccia della riga corrente. Inoltre viene passato, dalla funzione chiamante, il valore attuale di ogni impostazione corrente; in questo modo il programma sa quando stampare ON o OFF nelle stringhe delle funzioni di setting di questo sistema di controllo.

in pseudocodice:

```
receive_parameters();

print_menu_string(parameters.string[position]);

if (parameters.menu_voice.value == 1)
        print("ON");
else
        print("OFF");
```

Nello pseudocodice non è stato tenuto conto della situazione in cui si ricomincia a stampare le stesse stringhe (caso in cui si è arrivati all'ultima stringa del menù e si vuole stampare la successiva); semplicemente si ricomincia a contare da 1 e si stampa la prima stringa.

Il calcolo della riga da stampare è delegato alla funzione chiamante. Il passaggio di parametri viene eseguito tramite registri.

3.4 scelta e reset

Le funzioni di scelta (*scelta_backhome scelta_frecce_direzione* e *scelta_lockdoor*) e di reset (*reset_pressione*), che si trovano nei rispettivi file *scelta_backhome.s*, *scelta_frecce_direzione.s*, *scelta_lockdoor.s*, *reset_pressione.s*, hanno il compito di far scegliere all'utente l'impostazione desiderata (quindi ON, OFF e per le frecce direzione il numero lampeggii), oltrechè stampare il proprio sottomenù.

Funzionano tutte allo stesso modo: viene passato tramite registri il valore attuale dell'impostazione da modificare; tramite lettura di tastiera si ottiene il contrario dell'impostazione corrente (se è ON diventa OFF e viceversa) e viene restituito alla funzione chiamante il risultato da memorizzare. Per terminare il setting, basta premere il pulsante invio e si ritorna alla funzione chiamante. L'unica eccezione si fa per reset_pressione, in quando per effettuare il reset si preme il tasto '**r**', e per scelta_frecce_direzione.

in pseudocodice:

```
SCELTA:

print_actual_value();

loop:
read_char();

if (char has value in ('e', 'd'))
        actual_value = !actual_value;
        goto loop;

else
        set_and_exit();



RESET:

read_char();

if (char_red == 'r')
        reset_and_exit();

else
        exit();
```

La funzione scelta_frecce_direzione stampa a video l'impostazione attuale delle frecce direzione (il numero di lampeggii di default alla prima esecuzione del programma, poi i lampeggii selezionati), poi riceve in input il numero di lampeggii desiderati che deve essere compreso in un range tra 2 e 5; se viene inserito un numero più piccolo di 2 si imposta al minimo il numero di lampeggii (2), altrimenti se più grande di 5 si imposta al massimo (5).

in pseudocodice:

```

print_actual_value();

read_int();

if ( int_red > 5 )
        set(5);
        exit();

elif ( int_red < 2)
        set(2);
        exit();

else
        set(int_red);
        exit();
```

Capitolo 4

Funzioni generiche

Sono state inserite delle funzioni generiche che svolgono attività comuni come: lettura di una stringa, conversione di una stringa in intero, stampa a video. Queste funzioni sono rielaborazioni di funzioni viste a lezione e nelle esercitazioni (atoi.s, itoa.s, printf.s).

Queste sono riassunte in questo elenco:

- atoi: riceve una stringa in input e restituisce un intero;
- itoa: riceve in input un intero e restituisce una stringa da stampare;
- new_l: stampa a video il carattere \n;
- printf: stampa a video la stringa passata come parametro;
- print_string_error: stampa una stringa di errore.

L'unica funzione che necessita di una breve spiegazione è compare_key, che si trova nel file *compare_key.s*: questa riceve come parametro passato tramite registri un puntatore alla stringa appena letta, tramite il registro *ecx*; se questa stringa contiene un carattere di quelli ammessi dal sistema, allora lo restituisce alla funzione chiamante tramite il registro *edx*; altrimentri, sempre tramite lo stesso registro, restituisce il valore 1 che deve essere intrepretato come errore e in questo caso viene stampato un messaggio di errore.

4.1 Pseudo codice di tutto il programma

NB: è stato usato il carattere '#' per indicare una qualsiasi stringa; questo per evitare di scrivere codice ripetuto (esempio: activate_user_mode e activate_superuser_mode sono state sostituite con la stringa activate_#_mode perché in pseudocodice hanno lo stesso comportamento).

```
function main():

        check_stack();

        if (stack.numargs < 2)
                activate_user_mode();
        else
                if(stack.arg2 == password)
                        activate_superuser_mode();
                else
                        exit_with_error();
end;



function activate_#_mode():
        read_string();

        if (string.length  == 2)

                switch(string[0]){

                        case 'e':       menu(up, value[]);
                        case 'd':       menu(down, value[]);
                        case 'r':       menu(sub_menu, value[]);

                        default:        error();
                }


end;


function menu(position, value[]):

        if(position != sub_menu)
                print_menu_string(parameters.string[position]);

                if (value[position] == 1)
                        print("ON");
                else
                        print("OFF");

        else
                if (current_position == scelta#)
                        scelta();
                else
                        reset_pressione();

end;
```

```
function scelta#()

        print_actual_value();

        loop:
        read_char();

        if (char has value in ('e', 'd'))
                actual_value = !actual_value;
                goto loop;

        else
                set_and_exit();

end;


function_reset_pressione()

        read_char();

        if (char_red == 'r')
                reset_and_exit();

        else
                exit();
```

Capitolo 5

Scelte progettuali

È stato deciso di dividere il codice in più file possibili in modo da organizzare al meglio le possibili modifiche e la leggibilità; quindi in questo modo risulta più mantenibile il codice.

Il passaggio di parametri è sempre stato fatto tramite registri perché la quantità di parametri da passare era molto ridotta (al massimo 4 valori passati). In questo modo il codice risulta molto più veloce perché non si fanno accessi alla memoria; si sarebbe potuto usare lo stack per il passaggio dei parametri qualora ci fossero stati più parametri da passare alle funzioni chiamate.

È stato deciso inoltre di leggere i caratteri 'e','d','r' sia minuscoli che maiuscoli.