

# Una specie di «shell»



Realizzazione della shell .....	4566
Conclusione .....	4569

app.h 4566 gets.c 4565 kernel\_main.c 4569 shell.c 4566

Si conclude il lavoro del sistema giocattolo con una shell elementare, la quale deve acquisire i caratteri prodotti dalla tastiera e svolgere un compito in base al comando impartito. Ma prima di vedere il codice della funzione che svolge questo compito è necessario introdurre un'altra funzione, prevista dallo standard, che in precedenza è stata saltata: **gets()**, dichiarata nel file di intestazione 'stdio.h'.

La funzione **gets()** ottiene una stringa leggendo continuamente il contenuto della variabile 'os.kbd.key'.

Listato u174.1. './05/lib/gets.c'

```
#include <stdio.h>
#include <kernel/os.h>
char
*gets (char *s)
{
    int i;
    //
    // Legge os.kbd.char.
    //
    for (i = 0; i < 256; i++)
    {
        while (os.kbd.key == 0)
        {
            //
```

```

        // Attende un carattere.
        //
        ;
    }
    s[i] = os.kbd.key;
    os.kbd.key = 0;
    if (s[i] == '\n')
    {
        s[i] = 0;
        break;
    }
}
return s;
}

```

## Realizzazione della shell

<<

La shell è costituita dalla funzione *shell()*, dichiarata nel file di intestazione ‘app.h’, nel quale potrebbero essere inseriti i prototipi di altri tipi di applicazione, da avviare con l’aiuto della shell stessa.

Listato u174.2. ‘./05/include/app/app.h’

```

#ifndef _APP_H
#define _APP_H      1

void shell ();

#endif

```

## Listato u174.3. './05/app/shell.c'

```
#include <app/app.h>
#include <stdio.h>
#include <string.h>
#include <kernel/gdt.h>
#include <kernel/kernel.h>
#include <kernel/mm.h>
#include <kernel/multiboot.h>
void
shell (void)
{
    char command[256];
    //
    //
    //
    while (true)
    {
        printf ("# ");
        //
        // Legge un comando.
        //
        gets (command);
        //
        if (strcmp (command, "quit") == 0
            || strcmp (command, "q") == 0)
        {
            break;
        }
        else if (strcmp (command, "help") == 0
            || strcmp (command, "h") == 0)
        {
            printf ("shell commands:\n");
            printf ("h|help          = this help\n");
            printf ("q|quit          = quit the shell\n");
        }
    }
}
```

```

        printf ("i mb|info mb      = "
                "show multiboot info\n");
        printf ("i gdt|info gdt = show gdt\n");
        printf ("i mem|info mem = show memory map\n");
    }
else if (strcmp (command, "info mb") == 0
        || strcmp (command, "i mb") == 0)
    {
        mboot_show ();
    }
else if (strcmp (command, "info gdt") == 0
        || strcmp (command, "i gdt") == 0)
    {
        gdt_print (&os.gdtr);
    }
else if (strcmp (command, "info mem") == 0
        || strcmp (command, "i mem") == 0)
    {
        kernel_memory_show ();
        mm_list ();
    }
else
    {
        printf ("%s] unknown command: %s\n", __func__,
                command);
    }
}
}

```

La shell mostra un invito e si aspetta l’inserimento di comandi molto semplici, come ‘**i mem**’ per avere la mappa dell’utilizzo della memoria. Se si sbaglia non è possibile correggere e la pressione di tasti per la cancellazione provoca semplicemente la scrittura di codi-

ci non gestiti. Si osservi che anche gli spazi superflui contano come «errori».

## Conclusione

Per concludere viene mostrato il listato definitivo del file ‘kernel\_main.c’, in cui si avvia la shell. Se con questo sistema si volesse fare qualcosa di più, basterebbe intervenire nella shell stessa, senza ritoccare ulteriormente il file ‘kernel\_main.c’.

### Listato u174.4. ‘./05/kernel/kernel\_main.c’

```
#include <kernel/kernel.h>
#include <kernel/build.h>
#include <stdio.h>
#include <kernel/gdt.h>
#include <kernel/mm.h>
#include <stdlib.h>
#include <kernel/int.h>
#include <sys/syscall.h>
#include <kernel/timer.h>
#include <kernel/keyboard.h>
#include <app/app.h>
//
// Funzione principale, da dove si avvia il kernel.
//
void
kernel_main (unsigned long magic, multiboot_t *info)
{
    //
    // Inizializza i dati relativi alla gestione dello
    // schermo VGA, quindi ripulisce lo schermo.
    //
    vga_init ();
    clear ();
```

```

//
// Data e orario di compilazione.
//
printf ("05 %s\n", BUILD_DATE);
//
// Cerca le informazioni «multiboot».
//
if (magic == 0x2BADB002)
{
    //
    // Salva e mostra le informazioni multiboot.
    //
    mboot_info (info);
    mboot_show ();
    //
    // Raccoglie i dati sulla memoria fisica.
    //
    kernel_memory (info);
    //
    // Predispone la tabella GDT.
    //
    gdt ();
    //
    // Predispone la memoria libera per l'utilizzo.
    //
    mm_init ();
    //
    // Predispone il timer.
    //
    timer_freq (CLOCKS_PER_SEC);
    //
    // Predispone la tastiera.
    //
    keyboard_load ();
}

```

```
    echo ();
    //
    // Predispone la tabella IDT.
    //
    idt();
}
else
{
    printf ("%s] no \"multiboot\" header!\n",
            __func__);
}
//
// Shell.
//
shell ();
//
printf ("%s] system halted\n", __func__);
_Exit (0);
}
```

Nella schermata successiva si vede una breve interazione con la shell, dove appare anche un errore di digitazione.

```

# help
shell commands:
h|help          = this help
q|quit          = quit the shell
i mb|info mb    = show multiboot info
i gdt|info gdt  = show gdt
i mem|info mem  = show memory map
# info mb
[mboot_show] flags: 00000000000000000000000011111100111 mlow: 027F mhigh: 00007BC0
[mboot_show] bootdev: 00FFFFFF cmdline: "(fd0)/kernel"
# info gdt
[gdt_print] base: 0x0010E068 limit: 0x0017
[gdt_print] 0 00000000000000000000000000000000 00000000010000000001000000000000
[gdt_print] 1 0000000000000000000000001111011110000 000000000110000001001101000000000
[gdt_print] 2 0000000000000000000000001111011110000 000000000110000001001001100000000
# info em...
[shell] unknown command: info em...
# info mem
[kernel_memory_show] kernel 00100000..0010E5A4 avail. 0010E5A4..01EF0000
[kernel_memory_show] text 00100000..001058F4 rodata 00105900..0010600C
[kernel_memory_show] data 0010600C..0010600C bss 00106020..0010E5A4
[kernel_memory_show] limit 00001EF0
[mm_list] free 0010E5A8..01EF0000 size 01EEFFFC
# quit
[kernel_main] system halted

```