

Un programma per processare comandi: `smallsh`

Il programma `smallsh` è una versione semplificata della `shell` Unix.

Il programma `smallsh` legge comandi da standard input e li esegue in foreground o in background (non gestisce però altri meccanismi offerti dalla `shell`, quali l'espansione del pathname e la redirectione dell'I/O).

Lo schema del programma `smallsh` è il seguente:

```
while (EOF not typed)
{
    get command line from user
    assemble command args and execute
    wait for child    /* if foreground command */
}
```

La funzione `userin`

La funzione `userin` acquisisce la prossima linea di comando.

In particolare:

- stampa il prompt (passato come parametro);
- legge da standard input una linea di comando carattere per carattere, ritornando il controllo quando incontra un newline oppure end of file;
- memorizza la linea in un buffer globale;
- ritorna il numero di caratteri letti oppure EOF in caso di end of file.

Codice della funzione userin

```
#include "smallsh.h"

static char inpbuf[MAXBUF], tokbuf[2*MAXBUF],
    *ptr = inpbuf, *tok = tokbuf; /* buffer e puntatori globali */

int userin(char *p)
{
    int c, count;
    ptr = inpbuf;
    tok = tokbuf;

    printf("%s", p); /* stampa il prompt */

    count = 0;
```

... codice di userin

```
while(1)
{
    if ((c = getchar()) == EOF)
        return(EOF);
    if (count < MAXBUF)
        inpbuf[count++] = c;
    if ( c == '\n'  && count < MAXBUF)
    {
        inpbuf[count] = '\0';
        return count;
    }
    if (c == '\n') /* se linea troppo lunga, ricomincia */
    {
        printf("smallsh: input line too long\n");
        count = 0;
        printf("%s",p);
    }
}
}
```

II file smallsh.h

```
#include <unistd.h>
#include <stdio.h>
#include <sys/wait.h>

# define EOL          1  /* end of line */
# define ARG          2  /* normal arguments */
# define AMPERSAND    3
# define SEMICOLON    4

# define MAXARG       512 /* max. no. command arguments */
# define MAXBUF       512 /* max. length input line */

# define FOREGROUND   0
# define BACKGROUND   1
```

Le funzioni `procline`, `gettok` e `runcommand`

La funzione `procline` analizza una linea di comando, utilizzando la funzione `gettok`, e ricostruisce la lista degli argomenti.

Quando incontra un newline oppure un `;`, invoca la funzione `runcommand` per eseguire il comando.

La funzione `int gettok(char **outptr)`

estrae singoli “token” da una linea di comando. Un token è un nome di comando oppure un argomento.

La funzione `int runcommand(char **cline, int where)`

crea un processo che esegue il comando passato come primo parametro, in background o foreground, a seconda del valore del parametro `where`. In caso di esecuzione in foreground, la funzione `runcommand` aspetta la terminazione del comando.

Codice della funzione procline

```
#include "smallsh.h"

int procline(void)
{
    char *arg[MAXARG+1]; /* array di puntatori per runcommand */
    int toktype; /* tipo del token */
    int narg; /* numero di argomenti correnti */
    int type; /* FOREGROUND o BACKGROUND */

    narg = 0;

    for(;;) /* ciclo infinito */
    {
        switch(toktype = gettok(&arg[narg])){
        case ARG: if(narg < MAXARG)
                    narg++;
                break;
        }
```

... codice di procline

```
case EOL:
case SEMICOLON:
case AMPERSAND:
    if(toktype == AMPERSAND)
        type = BACKGROUND;
    else
        type = FOREGROUND;
    if(narg != 0)
    {
        arg[narg] = NULL;
        runcommand(arg, type);
    }
    if(toktype == EOL)
        return;

narg = 0;
break;
}
}
}
```

II main

```
#include "smallsh.h"
```

```
char *prompt = "Command> "; /* prompt */
```

```
main()
```

```
{
```

```
    while(userin(prompt) != EOF)
```

```
        procline();
```

```
}
```

Esercizio

Completare il programma `smallsh`, definendo il codice delle funzioni `gettok` e `runcommand`.

N.B.: se in fase di compilazione si ottiene un messaggio d'errore del tipo

```
smallsh.c:87: error: 'pid_t' undeclared (first use in this function)
```

aggiungere alle direttive `#include` del file `smallsh.h` la seguente:

```
#include <sys/types.h>
```