

Re: Command Line Interface (CLI): your recommendations

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2006-02/msg02783.html

- *From:* RSoIsCaIrLiIoA <zz@xxxx>
 - *Date:* Mon, 20 Feb 2006 09:44:40 +0100
-

do you like my CLI (Command Line Interface)? :))

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
```

```
#define UC unsigned char
#define U unsigned
```

```
#define BUFSIZ_ BUFSIZ
```

```
static const char *pc_pro = "CLI[\"vexit\" to end]> ";
```

```
struct you{
char* name;
char* pointer;
char* descrizione;
unsigned npar;
struct you* prev;
};
```

```
struct you *last=0;
```

```
/*limit sono i parametri + 1 (nome della funzione) */
int popola(char** v, char* buf, int limit);
```

```
int skip_line(FILE* pf)
{ int c; while( (c=fgetc(pf))!='\n' && c!=EOF ); return c;}
```

```
int cmp_from_now(char* a0, char* a1)
{ int i;
for( i=0; a0[i]!=0 && a0[i]==a1[i] && !isspace(a0[i]); ++i);
if( (isspace(a0[i])||a0[i]==0) && (isspace(a1[i])||a1[i]==0))
return 1;
else return 0;
```

Re: Command Line Interface (CLI): your recommendations

```
}  
  
int cerca(const char* name, U par)  
{ U i;  
  struct you *p=last, *v;  
  while(1) {if(p==0) break;  
  v=p; p=p->prev;  
  if( v->npar==par && strcmp(v->name, name)==0 )  
  return 1;  
  }  
  return 0;  
}
```

```
int  
inserisci(const char* name, char* pointer, U par, char*  
descrizione)  
{ struct you *tm;  
  char *tmp;  
  unsigned sz;  
  ///////////////////////////////////  
  if( name==0 || (sz=strlen(name))==0 )  
  return 0;  
  if( cerca(name, par)==1 )  
  return 0;  
  if( (tm =malloc(sizeof(struct you)))==0 )  
  return 0;  
  if( (tmp=malloc(sz+2))==0 )  
  {free(tm); return 0;}  
  tm->name=tmp; strcpy(tm->name, name);  
  tm->pointer= pointer; tm->npar=par;  
  tm->prev=last; last=tm;  
  tm->descrizione = descrizione;  
  return 1;  
}
```

```
void free_list(void)  
{ struct you *p=last, *v;  
  while(1) {if(p==0) break;  
  v=p; p=p->prev;  
  free(v->name); free(v);  
  }  
  last=0;  
}
```

```
void  
insert_f(char* name, char* pointer, char* par, char* descrizione)  
{ char *pn=0;  
  U paru=0;  
  int k;
```

Re: Command Line Interface (CLI): your recommendations

```
k=sscanf(pointer, "%p", &pn);
if(k!=1) {printf("parametro \"pointer\" non accettato\n"); return;}
k=sscanf(par , "%u", &paru);
if(k!=1) {printf("parametro \"parametri\" non accettato\n"); return;}
k=inserisci(name, pn, paru, descrizione);
if(k!=1){printf("problemi di memoria\n"); return;}
}
```

```
void sys(char* a1){if( system(a1)!=0 ) printf("\a"); }
void show(void){puts("cliShow() stub"); }
void version(void){puts("cliVersion() stub");}
void help(void)
{U i;
struct you *p=last, *v;
puts("Comandi(parametri)");
while(1) {if(p==0) break;
v=p; p=p->prev;
printf("%p %s(%u): %s\n",
(void*)(v->pointer), v->name, v->npar, v->descrizione);
}
}
```

```
void port(void){puts("cliPort() stub");}
void vexit(void)
{puts("cliExit() stub");}
free_list();
exit(0);
}
```

```
void add(char* a1, char* a2)
{int aa1=0, aa2=0;
sscanf(a1, "%d", &aa1);
sscanf(a2, "%d", &aa2);
printf("%d\n", aa1+aa2);
}
```

```
void sub(char* a1, char* a2)
{int aa1=0, aa2=0;
sscanf(a1, "%d", &aa1);
sscanf(a2, "%d", &aa2);
printf("%d\n", aa1-aa2);
}
```

```
int main(void)
{char buf[BUFSIZ_] = {0}, *a[32], *pc;
struct you *p;
int cv, led;
////////////////////////////////////
inserisci("sys",(char*) sys, 1, "chiama sistema");
```

Re: Command Line Interface (CLI): your recommendations

```
inserisci("show", (char*) show, 0, "mostra etc");
inserisci("version", (char*) version, 0, "mostra versione");
inserisci("help", (char*) help, 0, "help");
inserisci("h", (char*) help, 0, "help");
inserisci("-h", (char*) help, 0, "help");
inserisci("port", (char*) port, 0, "port");
inserisci("vexit", (char*) vexit, 0, "esce dal sistema");
inserisci("add", (char*) add, 2, "somma due numeri");
inserisci("sub", (char*) sub, 2, "fa differenza di due numeri");
inserisci("inserisci", (char*) insert_f, 4,
"inserisce una funzione:\ninserisci(name, pointer_to_fun,
parametri, descrizione)");
```

```
la:;
while (1)
{ printf("%s", pc_pro); fflush(stdout);
buf[BUFSIZ-2]=0; /* massimo BUFSIZ-1 chars */
if( fgets(buf, BUFSIZ, stdin) !=0 )
{ if(buf[BUFSIZ-2] !=0 && buf[BUFSIZ-2] != '\n')
{ printf("Linea troppo lunga\n");
if( feof(stdin) ) vexit();
if( skip_line(stdin) == EOF );
vexit();
goto la;
}
for(pc=buf; isspace(*pc); ++pc);
if( cmp_from_now(pc, "sys")==1 )
{ if(pc[3]==0) sys("\n");
else sys(pc+4);
goto la;
}
cv=popola(a, buf, 8);
if(cv<=0 || cv>8) goto la1;
else --cv;
for(p=last, led=0; led==0 && p!=0 ; p=p->prev)
{ if(strcmp(p->name, a[0])==0)
{ if(p->npar!=(U)cv)
{ la1:; printf("Parametri non corretti\n"); goto la; }
switch(p->npar)
{ case 0:
( (void (*)(void))(p->pointer) )();
led=1;
break;
case 1:
( (void (*)(char*))(p->pointer) )(a[1]);
led=1;
break;
case 2:
( (void (*)(char*, char*))(p->pointer) )(
a[1], a[2]);
led=1;

```

Re: Command Line Interface (CLI): your recommendations

```
break;
case 3:
( (void (*)(char*, char*, char*))(p->pointer) )
(a[1], a[2], a[3]);
led=1;
break;
case 4:
( (void (*)(char*, char*, char*, char*))
(p->pointer) ) (a[1], a[2], a[3], a[4]);
led=1;
break;
case 5:
( (void (*)(char*, char*, char*, char*, char*))
(p->pointer) ) (a[1], a[2], a[3], a[4], a[5]);
led=1;
break;
case 6:
( (void (*)(char*, char*, char*, char*, char*, char*))
(p->pointer) ) (a[1], a[2], a[3], a[4], a[5], a[6]);
led=1;
break;
case 7:
( (void (*)(char*, char*, char*, char*, char*, char*, char*))
(p->pointer) ) (a[1], a[2], a[3], a[4], a[5], a[6], a[7]);
led=1;
break;
default:
break;
} //end switch
} //if
} //for
if(led!=1) printf("funzione non presente\n");
} // if fgets
if(feof(stdin)) vexit();
} //while
return 0;
}
```

.