Systemnahe Software (SS 2001)

Abgabetermin: 21. Mai 2001

Beispiellösung

4 Umgebungsvariablen (3 Punkte)

Schreiben Sie *ohne Verwendung* der Bibliotheksfunktion `char *getenv()` ein C-Programm *var.c*, das zu einer als Parameter übergebenen Umgebungsvariable den passenden Wert zurückgibt, analog zu `echo`.

Beispiel:

```
thales$ echo $HOME
/home/thales/grabert
thales$ var HOME
/home/thales/grabert
thales$ var TTY
/dev/pts/35
thales$ var TTYRESET
2502:1805:bd:8a3b:3:1c:7f:15:4:0:0:0:11:13:1a:19:12:f:17:16:0:0:1
:1:0:00:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0
:0:0:0:0:0:0:0:0:0:0:0:0
```

Tipp: Über `main(int argc, char **argv, char **env)` bzw. `extern char **environ` kann auf die Umgebungsvariablen zugegriffen werden.

Lösung:

```
/*
 * Loesung zu Aufgabe 4 von Blatt 3
 * Thema: Umgebungsvariablen
 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```
char *mygetenv(char *name) {
   extern char **environ;
   int len = strlen(name);
   int i;

   for (i = 0; environ[i] != NULL; i++)
      if (strncmp(environ[i], name, len) == 0 && environ[i][len] == '=')
         return environ[i] + len + 1;
   return NULL;
}

int main(int argc, char **argv) {
   char *value;

   if (argc != 2) {
      fprintf(stderr, "usage: %s name\n", argv[0]);
      exit(1);
   }

   value = mygetenv(argv[1]);

   if (value)
      puts(value);
   else {
      printf("variable %s does not exist\n", argv[1]);
      exit(2);
   }

   return 0;
}
```

5 Once again, please: Tiny-Shell (7 Punkte)

Lösen Sie Aufgabe 1 von Blatt 1 nicht mehr mit `system()`, sondern mit Hilfe einer `execXX()`-Bibliotheksfunktion! Sie können sich eine passende Funktion der exec-Familie auswählen. Ihr Programm sollte in der Lage sein, Kommandos mit Positions-Parametern wie „`ls var.c`" oder „`ps -l -uswg`" auszuführen. Hintergrundkommandos (`&`) oder Dateiumlenkung (`< > |`) müssen nicht unterstützt werden. Der Sohnprozess sollte eine Fehlermeldung ausgeben, wenn er das angegebene Kommando nicht starten kann.

Tipp 1: Sparen Sie sich eine umständliche Prozessynchronisation. Der Vater wartet mit Hilfe von `int wait(int *stat);` auf den Sohn.
Beispiel:

```
void main()
```

```
{  int stat;

   switch (fork()) {
      case 0:
         puts("mir ist ploetzlich soo uebel ....");
         exit(1);
         break;
      default:
         wait(&stat);   /* wartet, bis Sohn gestorben ist! */
         puts("my son recently died ...");
   }
}
```

Tipp 2: Zerlegen Sie den Eingabestring der Shell mit Hilfe der Bibliotheksfunktion `strtok()` (siehe Manual), und bauen Sie dann mit den Ergebnissen den `argv[]`-Vektor für `execXX()` zusammen.
Beispiel:

```
void main()
{  char s[] = "Uihh: da sind aber viele Blanks drin!";
   char *hlp;
   extern char *strtok();

   hlp = strtok(s, " ");
   while (hlp) {
      puts(hlp);
      hlp = strtok((char*)0, " ");
   }
}
```

Lösung:

```
/*
 * Pseudo Microshell via System
 * Vater nimmt Kommando entgegen und Sohn fuehrt dieses via execl() aus
 * Prozesssynchronisation via wait
 */
#include <stdio.h>
#include <strings.h>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc, char **argv)
{  char buf[200];
   int pid;
   char *tok;
```

```
   char *args[20];
   int i = 0;
   int stat;

   printf("tinish > ");
   while (gets(buf) && strcmp(buf, "exit")) {
      tok = strtok(buf, " ");
      i = 0;
      while (tok) {
         args[i++] = strdup(tok);
         tok = strtok(NULL, " ");
      }
      args[i] = NULL;
      switch (pid = fork()) {
         case -1:   /* error */
            perror("fork()");
            exit(1);
            break;
         case 0:    /* Sohn */
            execvp(args[0], args);
            printf("tinish: cmd >%s< not found\n", args[0]);
            exit(0);
         default:
            wait(&stat);
      }
      printf("tinish > ");
   }

   return 0;
}
```