Systemnahe Software (SS 2001)

Abgabetermin: 28. Mai 2001

Beispiellösung

6 Objektorientiertes non-interaktives Schneckenrennen (8 Punkte)

Erbarmen! Die Schnecken sind wieder unterwegs. Sie fressen im Garten alles kahl, was sich ihnen in den Weg stellt ... dabei ist es doch viel cooler, wenn man sie zu Rennzwecken umerzieht ... (Schumi-Schnecke und so ...)
Schreiben Sie ein C-Programm, das ein Schneckenrennen zwischen n Rennschnecken simuliert. Jede Rennschnecke wird durch einen eigenen Prozess repräsentiert (also: objektorientiert :-)). Jede Schnecke zieht pro Zug zwischen 0 und 2 Felder weiter. Wie weit eine Schnecke zieht, soll sie selbst durch eine Zufallszahl bestimmen. Es gibt einen Rennleiter (Vaterprozess/ Scheduler/Schiedsrichter/Taktgeber), der das Rennen koordiniert: er gibt pro Sekunde jeder Schnecke ein Signal (im wahrsten Sinne des Wortes! benutzen Sie hier bitte *SIGUSR1*), damit sie ihren Zug ausführt. Die Schnecke, die zuerst am Ziel angekommen ist (nach k zurückgelegten Feldern), terminiert ihren Schneckenprozess. Der Vater checkt nach jeder Runde, ob bereits eine Schnecke terminiert hat (z.B. via *waitpid*) und kürt dann den Sieger. Sollten mehrere Schnecken gleichzeitig ans Ziel kommen, gewinnt die Schnecke mit der kleinsten Startnummer. Die restlichen Schnecken werden erbarmungslos via *SIGKILL* zur Schnecke gemacht ...

7 ASCII-getriebene Bildschirmdarstellung des Rennens (2 Punkte)

Damit die Sache etwas lustiger für die Fans am Streckenrand wird, werden wir das Rennen auf dem Bildschirm darstellen! Via *system("clear");* löscht man den Bildschirm. Jetzt kommt der überraschende Move: sendet man an ein Terminal (mit den meisten ;-) klappt es) die Sequenz *ESCAPE[y;xH* dann wird der Cursor genau an die Stelle (x,y) positioniert und eine Ausgabe via z.B. printf erfolgt ab dieser Stelle. Lassen Sie dadurch die Schnecken über den Bildschirm laufen! Zeichnen Sie durch den Spielleiter auch schon mal die Ziellinie auf den Schirm ... **Sie finden auf dem FTP-Server ein** *lauffähiges Demoprogramm*, **an dem Sie sich orientieren**

können!

Lösung:

```
/*
 * AI4 SS01 Blatt 4 (c) mg sai uni ulm 2001
 * objektiorientiertes non-interaktives Schneckenrennen via signale
 * ein Taktgeber (Vaterprozess) startet n Schnecken und gibt ihnen jeweils
 * nach 1s ein Signal zum Weiterlaufen
 */
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <wait.h>

#define MAXSNAILS       10         /* soviel Schnecken rennen   */
#define SIGCOMMUNICATE  SIGUSR1    /* ueber das Signal wird kommuniziert*/
#define GOAL            15         /* soweit laufen die Buben   */

int where_am_i;            /* soweit bin ich als Schnecke schon */
int who_am_i;              /* diese Schnecke bin ich   */

/*
 * diese Routine wird durch ein Signal vom Schiedsrichter (scheduler)
 * in der Schnecke aktiviert
 * die Schnecke loescht ihre alte Position auf dem Schirm, macht einen
 * Zufallszug zwischen 0 und 2 Zuegen und malt dann ihre neue Position
 * auf den Schirm
 */
void go_for_the_medal()
{  int move = (rand()%3);                    /* Zufallszug zwischen 0 und 2 */

   signal(SIGCOMMUNICATE, go_for_the_medal); /* Signal neu abfangen */
                                 /* alte Position loeschen */
   printf("\033[%d;%dH   ", where_am_i, who_am_i*4);

   where_am_i += move;                       /* neue Position ausgeben */
   printf("\033[%d;%dHs%02d", where_am_i, who_am_i*4, who_am_i);
   fflush(stdout);

   if (where_am_i >=GOAL)               /* bin am Ziel (oder schon weiter) */
      exit(0);                          /* HURRAAA   */
}

/*
```

```
 * eine Schnecke wird gestartet
 * initialisiert srand ueber die Zeit+Schneckennummer
 */

void schnecke(int snail)
{
   who_am_i = snail;                        /* global bekannt machen   */
   srand(time(0)+snail);                    /* random initialisieren   */
   signal(SIGCOMMUNICATE, go_for_the_medal); /* Signal abfangen   */

   printf("Schnecke %d gestartet\n", snail);
   while (1)                                /* warten auf Taktgeber      */
      sleep(1);
}

/*
 * Schiedsrichter und Taktgeber
 * der Taktgeber schickt jeder Schnecke ein Signal, dass sie weiterziehen soll
 * wartet dann 1s und prueft via waitpid ab, ob bereits eine Schnecke am Ziel
 * ist
 * haben mehrere Schnecken gleichzeitig das Ziel erreicht, gewinnt die Schnecke
 * mit der kleinsten Nummer
 */

void scheduler(int *snailpid)
{  int won = 0;                    /* hat schon eine Schnecke gewonnen?*/
   int i;
   int stat;                       /* Statusinfo des beendeten Prozess */

   sleep(1);                       /* warten bis alle bereit   */
   system("clear");                /* Bildschirm loeschen      */
   printf("\033[%d;%dHZIEL ", GOAL, 0);
   for (i=0; i< 75; i++)
      putchar('+');
   fflush(stdout);

   while (!won) {                  /* bis einer gewonnen hat   */
      for (i=0; i< MAXSNAILS; i++)  /* los! zieht!      */
         kill(snailpid[i], SIGCOMMUNICATE);

      sleep(1);                    /* da warten wir noch ein bisschen */
      printf("\r"); fflush(stdout);
      for (i=0; !won && i< MAXSNAILS; i++)         /* einer gewonnen?? */
         if (waitpid(snailpid[i], &stat, WNOHANG)==snailpid[i]) {
            won = i+1;
            printf("\n\nschnecke %d exit %d sig %d\n",
```

```
                  i+1, stat>>8, stat&0x7f);
         }
   }
   printf("Schnecke Nummer %d hat gewonnen. Herzlichen Glueckpunsch!\n",
                         won);
   for (i=0; i<MAXSNAILS; i++)      /* Restschnecken abraeumen   */
      if (i!=(won-1))
         kill(snailpid[i], 9);
}

int main()
{  int i;
   int snailpid[MAXSNAILS];               /* PIDs der Schnecken   */

   for (i=0; i< MAXSNAILS; i++) {         /* Schnecken erzeugen   */
      if ((snailpid[i]=fork())==0) {
         schnecke(i+1);
         exit(1);                         /* normally never reached ... */
      }
   }
   scheduler(snailpid);                   /* Schiedsrichter spielen */
   return 0;
}
```