|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБАРАТОРНОЙ РАБОТЫ №4**

по дисциплине: «Операционные системы»

|  |  |  |
| --- | --- | --- |
| Студент |  | Беркович Евгений Сергеевич |
| Группа |  | РК6-55Б |
| Тип задания |  | Конкуренция параллельных процессов |
| Вариант |  | 02 |

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_ **Беркович Е.С.**\_ \_

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_ **Грошев С. В.** \_\_

*подпись, дата фамилия, и.о.*

*Москва, 2023 г.*

**Техническое задание**

Требуется разработать программу, которая визуально демонстрирует поведение параллельных процесс os Unix на основе механизма диспетчеризации. С этой целью программа должна порождать заданное число процессов, которые должны выполняться параллельно, конкурируя за ресурсы процессора. При этом каждому порождённому процессу должна быть сопоставлена символьная метка в виде заглавной латинской буквы и номер столбца консоли, которая выбирается для её перемещения фиксированном диапазоне позиций. Число конкурирующих порожденных процессов и диапазон позиций для перемещения их буквенных меток должны передаваться в программу через аргументы командной строки её вызова Порожденные в программе процессы должны обеспечивать передвижение сопоставленных им буквенных меток по назначенным им строкам сверху вниз в заданном диапазоне позиций консоли, используя поток стандартного вывода.

**Приложение №1**

**Содержимое файла terminal.c**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <sys/timeb.h>  #include <sys/time.h>  #include <sys/types.h>  #include <sys/wait.h>  #include <unistd.h>  /\* clear standart text mode window \*/  void clrscr() {  unsigned char esc[11];  esc[0] = 27;  esc[1] = '[';  esc[2] = 'H';  write(1, esc, 3);  /\* obtain clear escape sequence \*/  esc[2] = '2';  esc[3] = 'J';  write(1, esc, 4);  return;  }  void gotoxy(int tx, int fixedcolumn, char c) {  unsigned char esc[16];  unsigned char ystr[3]; /\* vertical cursor location \*/  unsigned char fixed\_column[3]; /\* horizontal cursor location \*/  int i; /\* ESC-sequence current index \*/  int j; /\* cursor location current index \*/  /\* convert cursor location to text format \*/  if ((tx > 99) || (fixedcolumn > 99)) tx = fixedcolumn = 99;  if ((tx < 1) || (fixedcolumn < 1)) tx = fixedcolumn = 1;  fixed\_column[0] = fixed\_column[1] = fixed\_column[2] = '\0';  ystr[0] = ystr[1] = ystr[2] = '\0';  sprintf((char\*)ystr, "%d", tx - 1);  sprintf((char\*)fixed\_column, "%d", fixedcolumn);  /\* obtain goto escape sequence \*/  esc[0] = 27;  esc[1] = '[';  i = 2;  j = 0;  while (ystr[j]) esc[i++] = ystr[j++];  j = 0;  esc[i++] = ';';  while (fixed\_column[j]) esc[i++] = fixed\_column[j++];  esc[i++] = 'H';  // esc[i++]='\b';  esc[i++] = ' ';  esc[i] = '\0';  write(1, esc, i);  sprintf((char\*)ystr, "%d", tx);  i = 2;  j = 0;  while (ystr[j]) esc[i++] = ystr[j++];  j = 0;  esc[i++] = ';';  while (fixed\_column[j]) esc[i++] = fixed\_column[j++];  esc[i++] = 'H';  esc[i++] = c;  esc[i] = '\0';  write(1, esc, i);  return;  }  int main(int argc, char\* argv[]) {  int x = 1;  int status;  int i;  int j = 0;  int PROCNUM = atoi(argv[1]);  int\* pid = sbrk((PROCNUM + 1) \* sizeof(int));  char\* lead = sbrk((PROCNUM + 1) \* sizeof(char));  int dist = atoi(argv[2]);  int p;  char bell = '\007';  struct timeval tp;  int jump;  clrscr();  while (j < PROCNUM) {  if ((pid[j] = fork()) == 0) {  usleep(PROCNUM - j);  while (x < dist) {  gotoxy(x, j + 1, 'A' + j);  gettimeofday(&tp,NULL);  if (((tp.tv\_usec/1000)% (j + 'A')) != j) continue;  x++;  for (i = 0; i < 1000000; i++)  ;  }  exit('A' + j);  }  j++;  }  j = 0;  while ((p = wait(&status)) != (-1)) {  for (i = 0; i < PROCNUM; i++)  if (pid[i] == p) lead[j++] = (char)((status >> 8) & '\377');  write(1, &bell, 1);  }  lead[j] = '\n';  sleep(1);  gotoxy(dist + 3, 1, '\n');  write(1, lead, PROCNUM + 1);  exit(0);  } |

**Приложение №2**

**Результат работы программы:**

Программа принимает аргумент командной строки: количество процессов

, дистанция выполнения задачи для процесса.

Имена процессов движутся сверху вниз.

