|  |
| --- |
| #include <stdio.h> |
|  | #include <sys/stat.h> |
|  | #include <time.h> |
|  | #include <stdlib.h> |
|  | #include <unistd.h> |
|  | #include <fcntl.h> |
|  |  |
|  | const char\* NAME = "testfile"; |
|  |  |
|  | int number (const void\* a, const void\* b)//разница между числами |
|  | { |
|  | return (\*((int\*)b) - \*((int\*)a)); |
|  | } |
|  |  |
|  | int\* random(int n)//генерируем n случайных чисел |
|  | { |
|  | srand(time(NULL)); |
|  |  |
|  | int \*arr = malloc(sizeof(int) \* n); |
|  | for (int i = 0; i < n; ++i) |
|  | { |
|  | arr[i] = rand() % 30; |
|  | } |
|  |  |
|  | return arr; |
|  | } |
|  |  |
|  | void print(int\* arr, int n) //выводим на экран |
|  | { |
|  | for (int i = 0; i < n; ++i) |
|  | { |
|  | printf("%d ", arr[i]); |
|  | } |
|  | printf("\n"); |
|  | } |
|  |  |
|  | int main(int argv, char \*argc[]) |
|  | { |
|  | int n = atoi(argc[1]); |
|  | int \*arr = random (n); |
|  | print(arr, n); |
|  |  |
|  | mknod(NAME, S\_IFIFO | 0666, 0);// создает FIFO (именованный канал), специальный символьный или специальный блочный файл, с именем *имя*  ежим обозначается константой S\_IFIFO из заголовочного файла . Здесь же определяются права доступа. |
|  | int fd[2]; Неименованный канал создается вызовом pipe, который заносит в массив int [2] два дескриптора открытых файлов. fd[0] – открыт на чтение, fd[1] – на запись). Канал уничтожается, когда будут закрыты все файловые дескрипторы ссылающиеся на него. |
|  | pipe(fd);  Для дальнейшей работы с ним применяются системные вызовы *read()* , *write()* и *close()* . |
|  |  |
|  | int child\_id = fork();//порождение процесса |
|  |  |
|  | if (child\_id == 0)//порожден |
|  | { |
|  | close(fd[0]); //закрытие для чтения |
|  | int fifo = open(NAME, O\_RDONLY);// флаги описывают набор операций, которые, при успешном открытии файла, будут разрешены над файлом |
|  | int \*sndarr = malloc(sizeof(int) \* n);//выделяется блок памяти |
|  | read(fifo, sndarr, sizeof(int) \* n); |
|  | close(fifo); |
|  | qsort(sndarr, n, sizeof(int), number);//сортировка чисел по убыванию |
|  | write(fd[1], sndarr, sizeof(int) \* n);//запись чисел через неименованный |
|  | close(fd[1]); //закрытие для записи |
|  | } |
|  | else // не порожден |
|  | { |
|  | close(fd[1]); //закрытие для записи |
|  | int fifo = open(NAME, O\_WRONLY);//открываем fifo |
|  | write(fifo, arr, sizeof(int) \* n); |
|  | close(fifo); |
|  | int \*sortarr = malloc(sizeof(int) \* n); |
|  | read(fd[0], sortarr, sizeof(int) \* n); |
|  | print(sortarr, n);//процесс выводит на экран отсортированный массив |
|  | close(fd[0]); //закрытие для чтения |
|  | unlink(NAME); удаляет имя из файловой системы |
|  | } |
|  | } |