Операционные системы

лекции 5, 6

- Примеры системных вызовов: низкоуровневая работа с файлами.

- Создание временных файлов

- Перенаправления ввода/вывода.

1.11.2010

Системные вызовы для работы с файлами

Дескриптор
ТОФ (Таблица Открытых Файлов)

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int open(const char *pathname, int flags);
int open(const char *pathname, int flags,
                               mode_t mode);
```

`O_RDONLY` `O_WRONLY` `O_RDWR`

`O_CREAT` `O_EXCL`

`O_TRUNC` `O_APPEND`

Системные вызовы для работы с файлами

```
int creat(const char *pathname, mode_t mode);
```

```
int fd;
fd = open("f", O_WRONLY|O_CREAT|O_TRUNC,0644);

int fd;
fd = creat("f",0644);
```

```
#include <unistd.h>
int close(int fd);
```

Системные вызовы для работы с файлами

Чтение

```
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);
```

Запись

```
#include <unistd.h>

ssize_t write(int fd, void *buf, size_t count);
```

Пример

Что делает следующий фрагмент программы?

```
int n;
char buf[256];

while(n=read(0,buf,256)>0)
 write(1,buf, 256);
```

Пример

Исправленный вариант

```
int n;
char buf[256];

while((n=read(0,buf,256))>0)
 write(1,buf, n);
```

Задача

Написать функцию

```
int getchar(void)
```

которая осуществляет небуферизованный ввод, читая по одной литере из входного потока

Варианты решения

```
a) int getchar (void){
      char c;
      return ( read (0, &c, 1) == 1) ? c : EOF);
   } /*неверно*/
```

```
b) int getchar (void){
      int c;
      return ( read (0, &c, 1) == 1) ? c : EOF);
   }/*неверно*/
```

```
c) int getchar (void){
    char c;
return (read (0, &c, 1) == 1) ? (unsigned char)c : EOF);
   }
```

```
d) int getchar (void){
    char c;
    return ( read (0, &c, 1) == 1) ? (int)c : EOF);
   }/*неверно*/
```

Системные вызовы для работы с файлами

Позиционирование

```
#include <sys/types.h>
#include <unistd.h>
off_t lseek(int fd, off_t offset, int whence);
```

```
#define SEEK_SET 0
#define SEEK_CUR 1
#define SEEK_END 2
```

Задача

В переменную L записать длину (размер) файла "data.dat" в байтах.

Системные вызовы для работы с файлами

В переменную L записать длину (размер) файла "data.dat" в байтах.

```
int main(){
  int fd;
  long L;
  fd = open("data.dat",O_RDONLY);
  if (fd==-1) {
      perror("open");
      return 1;
  }
  L=lseek(fd,0, SEEK_END);

  write(1,&L, sizeof(long));
  return 0;
}
```

Файловые дескрипторы и дескрипторы потоков

```
#include <stdio.h>
int fileno(FILE *stream);
```

```
#include <stdio.h>
FILE *fdopen(int fildes, const char *mode);
```

Перенаправления ввода - вывода

Командная строка shell

```
./a.out <f.dat >output.dat 2>err.dat

./a.out <f.dat >>output.dat

./a.out | ./b.out
/* реализация конвейера будет рассмотрена на
следующих лекциях*/

./a.out 2 >& 1 | ./b.out
```

Перенаправления ввода-вывода

int dup (int fd);
по заданному дескриптору файла fd будет возвращен дескриптор файла с наименьшим свободным номером, в ТОФ будет в строку этого дескриптора скопирована информация из строки, соответствующей fd.
В случае ошибки функция dup возвращает -1.

int dup2 (int fd1, int fd2);
дескриптор fd2 будет указывать на тот же файл, что и дескриптор fd1. Если с дескриптором fd2 уже был связан открытый файл, то он предварительно закроется.
В случае ошибки функция dup2 возвращает -1.

---

ulimit -a или ulimit -n
Команды, позволяющие узнать максимально допустимое число одновременно открытых дескрипторов

Пример

```
int main(int argc, char** argv){
  int fd,N;
  if (argc<2)return 1;
  fd = open (argv[1],O_RDONLY);
  if (fd==-1){
    perror("open");
    return 1;
  }
  close(0);
  dup(fd);
  close(fd);
  scanf("%d",&N);
  printf("N=%d\n",N);
  return 0;
}
```

Пример

```
int main(int argc, char** argv){
  int fd;
  if (argc<2)return 1;
  fd = open (argv[1],O_WRONLY);
  if (fd==-1){
    perror("open");
    return 1;
  }
  dup2(fd,1);
  close(fd);

  printf("Hello, world\n");

  return 0;
}
```

Примеры

```
cat mytext.ps>/dev/lp0
```

```
cat /dev/cdrom > image.iso
```

Домашнее задание 3 (срок сдачи - 3.11.2010)

Используя низкоуровневый ввод-вывод, реализовать две функции

```
int puts (const char *s);

int getchar (void);
```

(Написать буферизованный вариант функции int getchar(void), когда функция осуществляет ввод большими порциями, но при каждом обращении к ней выдает только одну литеру.)