

Самарский национальный исследовательский университет им. акад. С.П. Королева

ОПЕРАЦИОННЫЕ СИСТЕМЫ СЕМЕЙСТВА UNIX

Часть IV

Составитель: к.т.н., доц. К.Е. Климентьев

Самара 2015

1. Дисковые файлы и файлы устройств

Файл – именованный набор данных все зависимости от его месторасположения и формы представления.
Базовые операции над файлом: 1) чтение; 2) запись. Вспомогательные операции: перемещение указателя.
Служебные операции: 1) открытие; 2) закрытие.

Примеры:

- /etc/passwd – дисковый файл (описание пользователей);
- /dev/sda – носитель или /dev/hda1 – раздел носителя;
- /dev/tty1 – консоль, /dev/ttys0 – com-порт и пр.;
- /dev/mem – физическая память;
- /dev/random – ДСЧ, /dev/urandom – ДПСЧ;
- /dev/null, /dev/zero, /dev/full и пр. – «спецфайлы»;
- канал (будет дальше);
- сокет (будет дальше).

1.1. Архитектура UNIX



2. «Буферизованный» или «стандартный» ввод-вывод

Буфер располагается в адресном пространстве (в хипе или стеке) задачи. Основные функции:

- `fopen()` и `fclose()` – открыть и закрыть файл;
- `fread()` и `fwrite()` – читать и писать файл;
- `fseek()` – перемещение указателя в файле;
- `fflush()` – форсировать передачу из буфера и в буфер;
- `fgetc()`, `fputc()`, `fgets()`, `fputs()`, `fprintf()`, `fscanf()` – операции форматированного ввода-вывода;
- `getc()`, `putc()`, `gets()`, `puts()`, `printf()`, `scanf()` – операции форматированной работы с консолью;
- `sgetc()`, `sputc()`, `sgets()`, `sputs()`, `sprintf()`, `sscanf()` – форматированная работа со строкой.

```
#include <stdio.h>
main() { // Побайтовое копирование файла описания пользователей
    FILE *f1, *f2; unsigned char c;
    f1 = fopen("/etc/passwd", "r+b"); // Readonly + binary
    f2 = fopen("./file.txt", "w");
    while (1) {
        if (feof(f1)) break;
        fread( &c, sizeof(unsigned char), 1, f1 );
        fwrite( &c, sizeof(unsigned char), 1, f2 );
        fprintf(stdout, "%c", c); // stdin, stdout, stderr
    }
    fclose(f1); fclose(f2);
}
```

3. «Простой» или «базовый» ввод-вывод

Основные функции:

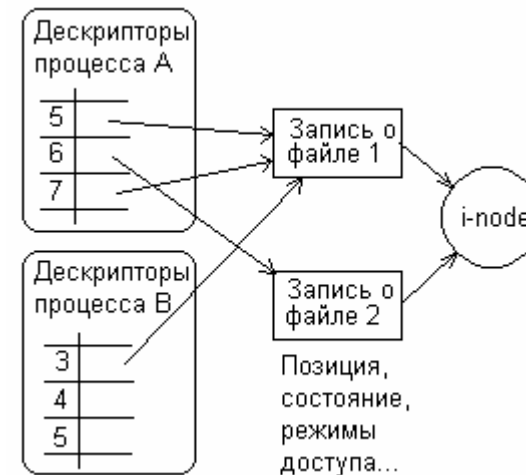
- `int open (const char * filename, int flags, mode_t mode)` – открытие или создание файла; флаги в `fcntl.h`: `O_RDONLY`, `O_WRONLY`, `O_RDWR`, `O_APPEND`, `O_TRUNC`, `O_CREAT` и пр.; режимы доступа в `/sys/stat.h`: `S_IRREAD`, `S_IWRITE`, `S_IXEXEC` и прочие биты.
- `int creat(const char *filename, mode_t mode)` эквивалентно `int open(..., O_WRONLY|O_CREAT|O_TRUNC,...)`;
- `ssize_t read (int fd, void * buffer, size_t count)` – чтение файла;
- `ssize_t write (int fd, const void * buffer, size_t count)` – запись в файл;
- `int close (int fd)` – закрытие файла;
- `off_t lseek (int fd, off_t offset, int against)` – перемещение указателя в файле, режимы в `fcntl.h`: `SEEK_SET=0`, `SEEK_CUR=1`, `SEEK_END=2`;
- `pread()` и `pwrite()` – чтение и запись без перемещения указателей.

```
#include <stdio.h> #include <fcntl.h>
#include <errno.h>
```

```
main() { // Чтение даты создания BIOS
    int r, f, i; unsigned char c;
    f = open("/dev/mem", O_RDONLY, 0);
    lseek(f, 0xFFFFF5, SEEK_SET);
    for (i=0; i<8; i++) {
        r = read(f, &c, sizeof(char));
        write(1, &c, sizeof(char)); // STDIN_FILENO=0,

        STDOUT_FILENO=1, STDERR_FILENO=2
    }
    close(f);
}
```

3.1. Файловые дескрипторы

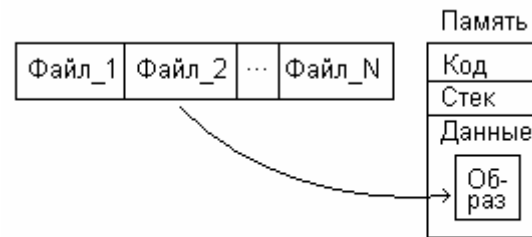


4. Проецирование файлов в память

Основные функции:

- `mmap()` – отображение файла в память, доступ как к массиву байтов;
- `munmap()` – отмена отображения.

4.1. Принцип отображения



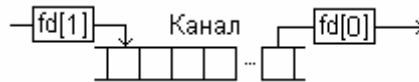
```
#include <stdio.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h> main() {
    int r, f, i; char *m;
    f = open("/dev/mem", O_RDONLY, 0);
    m = (char *) mmap( 0, 0xFFFFF, PROT_READ, MAP_SHARED, f, 0);
    for (i=0xFFFF5; i<0xFFFFD; i++) write(1, &m[i], 1);
    munmap( m, 0xFFFFF);
    close(f);
}
```

5. Каналы ввода-вывода (“пайпы”)

Каналы (пайпы) – файлы с последовательным доступом. Назначение: IPC. Функции:

- pipe() – создать и открыть неименованный канал с 2 дескрипторами: для ввода и вывода;
- mkfifo() – создать именованный канал;
- open(), read(), write() – назначение таково же, как для обычных файлов.

5.1. Принцип доступа



```
#include <stdlib.h> #include <stdio.h>
```

```
int pd[2]; // Pipe descriptors
```

```
int main() {
```

```
    int d1=1234, d2;
```

```
    pipe(pd); // create pipe
```

```
    write(pd[1], &d1, sizeof(int));
```

```
    read(pd[0], &d2, sizeof(int));
```

```
    printf("%u", d2);
```

```
    close(pd[0]); close(pd[1]);
```

```
}
```

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

```
#include <fcntl.h>
```

```
#include <string.h>
```

```
#include <stdio.h>
```

```
main () {
```

```
    int f1, f2, d1=1234, d2, q;
```

```
    q=mkfifo("/tmp/tralala", 0x777); // Create
```

```
    f2 = open("/tmp/tralala",
```

```
    O_RDONLY|O_NONBLOCK); // !!!
```

```
    f1 = open("/tmp/tralala", O_WRONLY);
```

```
    q = write(f1, &d1, sizeof(int));
```

```
    close(f1);
```

```
    q = read(f2, &d2, sizeof(int));
```

```
    close(f2);
```

```
    printf("%u", d2);
```

```
}
```

6. Ввод-вывод

6.1. Модели ввода-вывода



Модели ввода-вывода:

- 1) Блокирующий синхронизированный;
- 2) Блокирующий синхронный;
- 3) Асинхронный.

Модели оповещения о завершении асинхронного ввода-вывода:

- A) проверка признака готовности;
- B) ожидание готовности;
- C) обработка сигнала готовности SIGIO.

7. Мультиплексированный ввод-вывод

Функции и макросы:

- `select(maxdesc, fd_set *rd, fd_set *wd, fd_set *ed, timeval *t)` – ожидание завершения + разрушение списка;
- `select(maxdesc, fd_set *rd, fd_set *wd, fd_set *ed, timeval *t, sigset_t *sigmask)` + реакция на сигналы.
- `FD_ZERO()` – очищает список;
- `FD_SET()` – добавляет к списку дескриптор;
- `FD_CLR()` – удаляет дескриптор;
- `FD_ISSET()` – проверяет готовность дескриптора.

```
#include <stdio.h>
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>
main() {
    int len, ret;
    fd_set readfds; // Список дескрипторов ввода
    struct timeval tv; // Параметры тайм-аута
    unsigned char buf[16];
    FD_ZERO(&readfds); // Очистка списка
    FD_SET( STDIN_FILENO, &readfds); // Добавление операции ввода
    tv.tv_sec=5; // Тайм-аут ожидания
    tv.tv_usec=0; // Микросекунды
    ret = select(STDIN_FILENO+1, &readfds, NULL, NULL, &tv); // Ждем хотя бы одного
    if (ret==0) {
        printf("\nGame over\n");
    }
    else {
        len=read(STDIN_FILENO, buf, 16);
        write (STDOUT_FILENO, buf, len);
    }
}
```


8. Асинхронный ввод-вывод

Функции:

- `aio_read()` – инициализация ввода;
- `aio_write()` – инициализация вывода;
- `lio_listio()` – инициализация списка операций;
- `aio_error()` – возвращает статус асинхронной операции;
- `aio_fsync()` – ждет завершения всех операций вывода;
- `aio_cancel()` – прерывает незавершенную операцию ввода-вывода.

```
#include <sys/types.h> // gcc example.c -o example -lrt
#include <aio.h>
#include <fcntl.h>
#include <errno.h>
    struct aiocb cb;
int number, f;

int main() {
    f = open("/dev/random", O_RDONLY, 0);
    memset(&cb, 0, sizeof(struct aiocb));
    cb.aio_nbytes = sizeof(int);
    cb.aio_fildes = f;
    cb.aio_offset = 0;
    cb.aio_buf = &number;
    aio_read(&cb);
    while(aio_error(&cb) == EINPROGRESS); // Ждем
    printf("0x%X", number);
}
```

9. Файловая система

Файловые системы UNIX: ext*fs (Linux), HFS и HFS+ (Mac OS X), UFS (BSD) и пр.

9.1. Диск

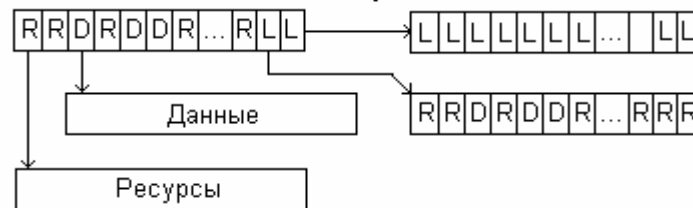
Суперблок i-nodes Данные



9.2. i-node

#	Атрибуты	Размер	Список кластеров	Счетчик
---	----------	--------	------------------	---------

9.3. Список кластеров



9.4. Каталоги

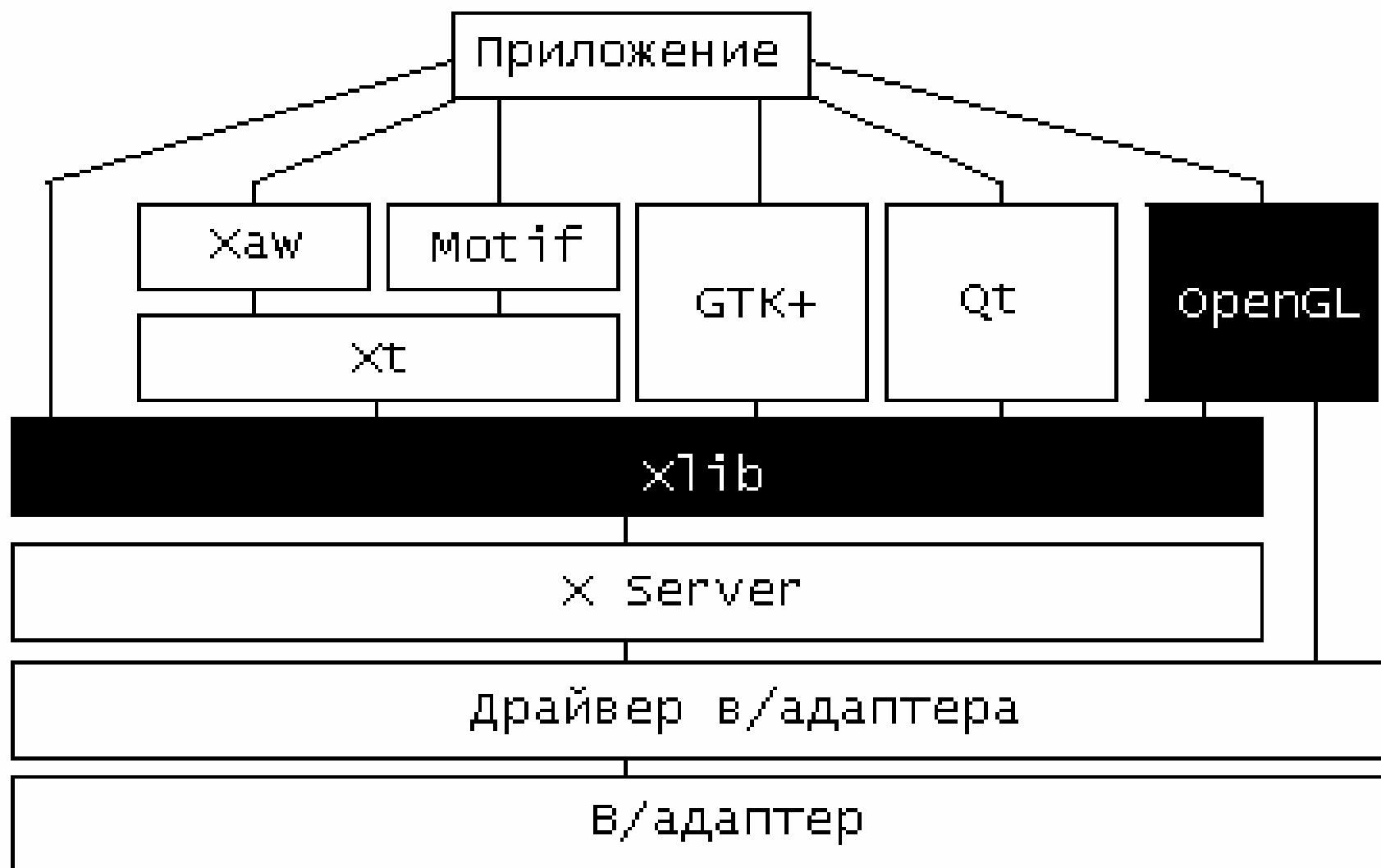
#	Имя
4	myfile
⑤	vasya
6	masha

#	Имя
7	tanya
⑤	pupkin

9.5. Файлы ссылок

masha - символическая ссылка
⑥ masha - псевдоним (alias).

10. Видеоподсистема

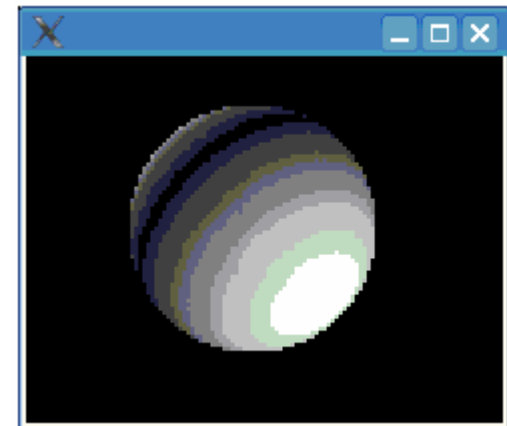
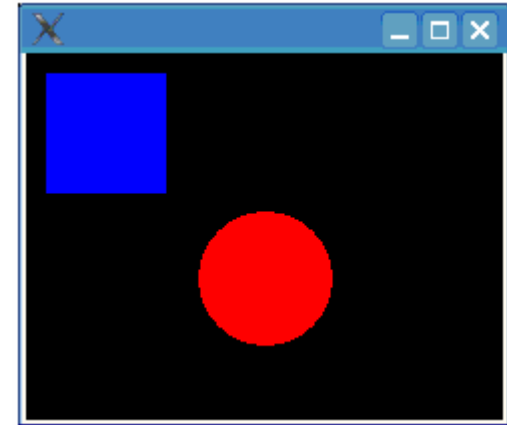


11. Низкоуровневая графика средствами XLib

```
// gcc myx.cpp -L/usr/X11R6/Lib -lX11
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
#include <unistd.h>
#define WIN_WIDTH 640
#define WIN_HEIGHT 480
unsigned long GetColor( Display* dis, char* color_name ) {
    Colormap cmap; XColor near_color, true_color;
    cmap = DefaultColormap( dis, 0 );
    XAllocNamedColor( dis, cmap, color_name, &near_color, &true_color );
    return( near_color.pixel );
}
int main( void ) {
    Display* dis; Window win; XSetWindowAttributes att; GC gc; XEvent ev; int t;
    dis = XOpenDisplay( NULL );
    win = XCreateSimpleWindow( dis, RootWindow(dis,0), 100, 100,
                               WIN_WIDTH, WIN_HEIGHT, 5, WhitePixel(dis,0), BlackPixel(dis,0) );
    att.backing_store = WhenMapped;
    XChangeWindowAttributes( dis, win, CWBackingStore, &att );
    XSelectInput( dis, win, ExposureMask );
    XMapWindow( dis, win );
    do { XNextEvent( dis, &ev); } while( ev.type != Expose );
    gc = XCreateGC( dis, DefaultRootWindow(dis), 0, 0 );
    XSetFunction( dis, gc, GXxor );

    XSetForeground( dis, gc, BlackPixel(dis,0)^GetColor( dis, "blue" ));
    XFillRectangle( dis, win, gc, 10, 10, 60, 60 );
    XSetForeground( dis, gc, BlackPixel(dis,0)^GetColor( dis, "red" ));
    XFillArc( dis, win, gc, 70, 70, 50, 50, 0, 360*64);

    for (t=0;t<10;t++) sleep(1);
    XDestroyWindow( dis , win ); XCloseDisplay( dis ); return(0);
}
```



12. Структура OpenGL

4.1. Каркас программы

```
// gcc example.c -o example -lGL -lglut
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

// Обработка отображения
void Display() { ... }
// Обработка изменений размеров окна
void Reshape() { ... }
// Обработка клавиш
void Keyboard() { ... }

main(int argc, char *argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB);
    glutInitWindowSize(320, 200);
    glutCreateWindow("Header of window");

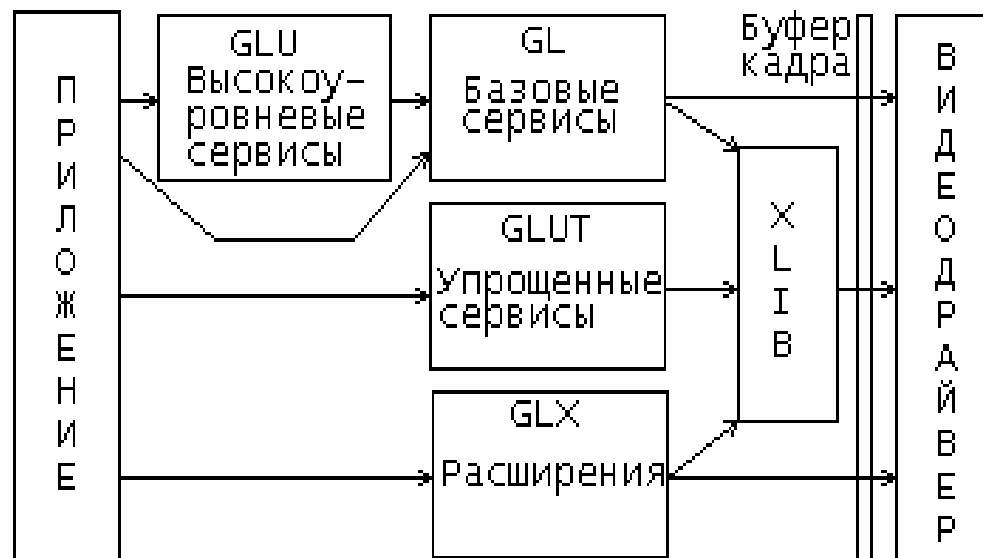
    glutDisplayFunc(Display);
    glutReshapeFunc(Reshape);
    glutKeyboardFunc(Keyboard);

    glutMainLoop();
}
```

4.2. Библиотечные каталоги

Linux/BSD	Mac OS X
GL/gl.h	OpenGL/gl.h
GL/glu.h	OpenGL/glu.h
GL/glut.h	GLUT/glut.h

4.3. Структура библиотек



13. GLUT – рисование по вершинам

Основные функции:

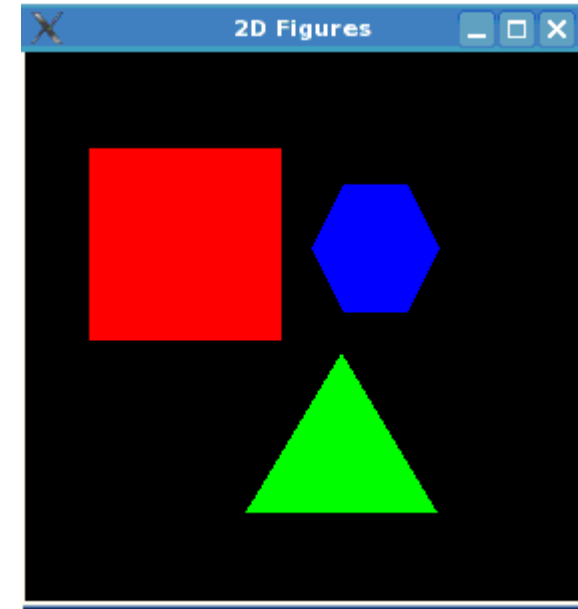
- `glBegin()` и `glEnd()` – начало и конец блока описания вершин;
- `glColor2/3/4/s/i/f/d()` – установка цвета вершины;
- `glVertex2/3/4/s/i/f/d()` – описание координат вершины.

Основные типы: `GL_POINTS`, `GL_LINES`, `GL_TRIANGLES`, `GL_QUADS`, `GL_POLYGON`.

Пример:

```
// gcc glfig.c -o glfig -lGL -lglut
```

```
void display() {  
    glClear(GL_COLOR_BUFFER_BIT);  
  
    glBegin(GL_QUADS);  
    glColor3f(1.0f, 0.0f, 0.0f); // Color = RED  
    glVertex2f(-0.8f, 0.1f); glVertex2f(-0.2f, 0.1f);  
    glVertex2f(-0.2f, 0.7f); glVertex2f(-0.8f, 0.7f);  
    glEnd();  
  
    glBegin(GL_TRIANGLES);  
    glColor3f(0.0f, 1.0f, 0.0f); // Color = GREEN  
    glVertex2f(0.1f, -0.6f); glVertex2f(0.7f, -0.6f); glVertex2f(0.4f, -0.1f);  
    glEnd();  
  
    glBegin(GL_POLYGON);  
    glColor3f(0.0f, 0.0f, 1.0f); // Color = BLUE  
    glVertex2f(0.4f, 0.2f); glVertex2f(0.6f, 0.2f); glVertex2f(0.7f, 0.4f);  
    glVertex2f(0.6f, 0.6f); glVertex2f(0.4f, 0.6f); glVertex2f(0.3f, 0.4f);  
    glEnd();  
    glFlush(); // Render now  
}
```



14. GLUT – рисование 3-мерных объектов

Основные функции:

`gluNewQuadric()` – создание нового 3D-объекта;

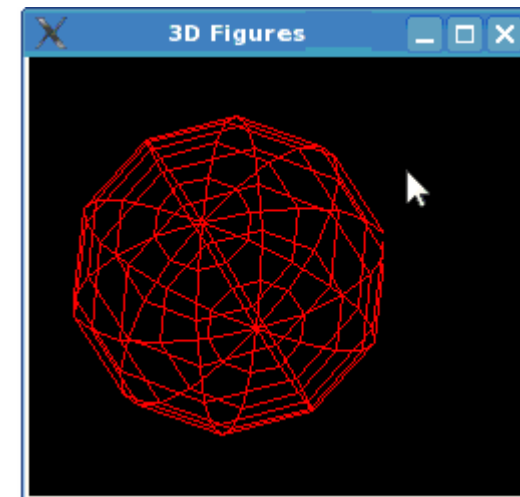
`gluSphere()` и `gluCylinder()` – определение объектов;

`glPushMatrix()` и `glPopMatrix()` – сохранение и восстановление матриц вершин;

`glTranslate()`, `glRotate()` и `glScale()` – сдвиг, поворот и масштабирование.

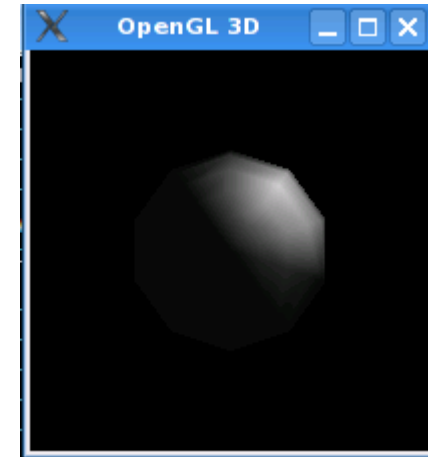
Пример:

```
void display() {  
    GLUQuadricObj *quadObj;  
    quadObj = gluNewQuadric();  
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );  
    glColor3d(1,0,0);  
    gluQuadricDrawStyle(quadObj, GLU_LINE);  
    gluSphere(quadObj, 0.5, 10, 10);  
    glRotated(20, 1, 0, 0);  
    glRotated(10, 0, 1, 0);  
    glRotated(30, 0, 0, 1);  
    glFlush();  
}
```



15. GLU/GLUT – рисование реалистичных 3D-сцен

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
void Display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Z-Buffer
    turn on
    GLUquadricObj *qobj;
    qobj=gluNewQuadric();
    glPushMatrix();
    gluSphere(qobj,0.5,10,10);
    glPopMatrix();
    glutSwapBuffers();
    glFlush();
}
int main(int argc,char **argv) {
    // float pos[3]={-0.8,0.5,0}; // Lamp position
    float pos[3]={-0.8, -0.8, 0.8};
    float dir[3]={0,0,0}; // Direction of lamp vector
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(200,200);
    glutInitWindowPosition(100,0);
    glutCreateWindow("OpenGL 3D");
    glClearColor(0.,0.,0.,1);
    glutDisplayFunc(Display);
    glEnable(GL_LIGHTING); // Enable lighting
    glEnable(GL_LIGHT0); // Turn on the lamp 0
    glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,dir); // Register direction
    glLightfv(GL_LIGHT0,GL_POSITION,pos); // Register position
    glutMainLoop();
}
```

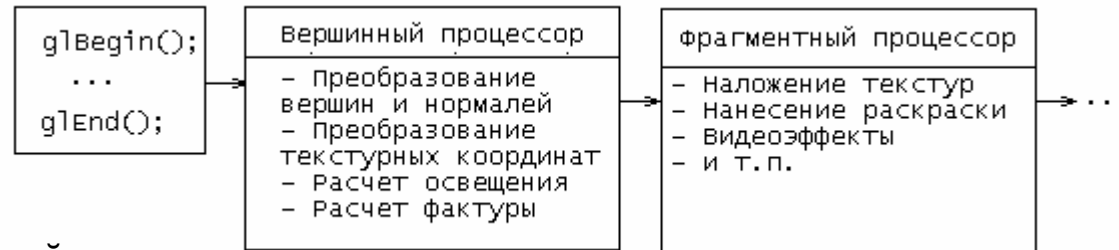


Параметры:

- 10 граней
- Освещение по Фонгу
- Материал – матовый

16.Шейдеры

11.1. Фрагмент графического конвейера в OpenGL



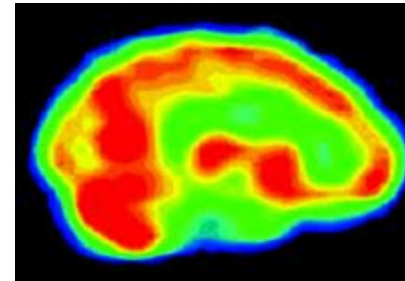
11.2. Языки шейдеров



11.3. Пример пары шейдеров на GLSL

```
// Вершинный - выполняется для каждой вершины
uniform float CoolestTemp;
uniform float TempRange;
attribute float VertexTemp;
varying float Temperature;
void main() {
    Temperature = (VertexTemp - CoolestTemp) / TempRange;
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}

// Фрагментный - выполняется для каждого фрагмента
uniform vec3 CoolestColor;
uniform vec3 HottestColor;
varying float Temperature;
void main() {
    vec3 color = mix(CoolestColor, HottestColor, Temperature);
    gl_FragColor = vec4(color, 1.0);
}
```



17. Примеры OpenGL-графики

