

Summary of Lesson 16

Functions , introduced in this lesson

```
t_ed *ed_new (t_object *assoc);
// getting a new editor

void ed_settext (t_ed *ew, Handle textHandle, long textSize);
// write text in editor window

void *connection_client (void *client, t_symbol *name, t_symbol *class,
method traverse);
// establish client connection

void connection_server (void *server, t_symbol *name);
// establish server connection

void connection_send (void *server, t_symbol *name, t_symbol *message,
void *arg);
// sending message

void connection_delete (void *obj, t_symbol *name);
// notify deleted

long evnum_get (void);
// get event number

Expr *expr_new (short argc, t_atom *argv, t_atom *types);
// expr object in an object

void expr_eval (Expr *ex, short argc, t_atom *argv, t_atom *result);
// evaluate evpr calculation
```

ex1.c

```
#include "ext.h"

// text editor

typedef struct {
    t_object b_ob;
    t_ed *ed;
} t_ex1;

void *ex1_new(void);
void *ex1_edsave (t_ex1 *x, char **text, long size, char *filename, short vol);
void ex1_edclose (t_ex1 *obj, char **text, long size);

void *ex1_class;
```

```

void main(void)
{
    setup((t_messlist **)&ex1_class, (method)ex1_new, 0L, (short)sizeof(t_ex1),
0L, 0L);
    address ((method)ex1_edclose, "edclose", A_CANT, 0);
    address ((method)ex1_edsave, "edsave", A_CANT, 0);
}

void *ex1_edsave (t_ex1 *x, char **text, long size, char *filename, short vol)
{
    post("file was saved",*text);
    post("size:%d",size);
    post("file:%s,location:%d",filename,vol);

    return 1; //not saved
    //return 0; saved
}

void ex1_edclose (t_ex1 *obj, char **text, long size)
{
    int i=0,j=0;
    char tempbuffer[256];
    post("window closed");
    post("size:%d",size);

    for(i = 0;i<size;i++)
    {
        if(i == size-1)
        {
            tempbuffer[j] = *((*text)+i);
            tempbuffer[j+1] = '\0';
            post("%s",tempbuffer);
            break;
        }
        else if(13 == *((*text)+i))
        {
            tempbuffer[j] = '\0';
            post("%s",tempbuffer);
            sprintf(tempbuffer, " "); //clear
            j = 0;
        }
        else
        {
            tempbuffer[j] = *((*text)+i);
            j++;
        }
    }

}

}

```

```

void *ex1_new(void)
{
    t_ex1 *x;
    x = (t_ex1*)newobject(ex1_class);
    x->ed = ed_new((t_object*)x);
    return x;
}

```

ex2.c

```

#include "ext.h"

// double click after close

typedef struct {
    t_object b_ob;
    t_ed *ed;
} t_ex2;

void *ex2_new(void);
void *ex2_edsave (t_ex2 *x, char **text, long size, char *filename, short vol);
void ex2_edclose (t_ex2 *obj, char **text, long size);
void ex2_eddbclick(t_ex2 *x);

void *ex2_class;

void main(void)
{
    setup((t_messlist **)&ex2_class, (method)ex2_new, 0L, (short)sizeof(t_ex2),
0L, 0L);
    address ((method)ex2_edclose, "edclose", A_CANT, 0);
    address ((method)ex2_edsave, "edsave", A_CANT, 0);
    address ((method)ex2_eddbclick, "dblclick", A_CANT, 0);
}

void *ex2_edsave (t_ex2 *x, char **text, long size, char *filename, short vol)
{
    post("file was saved",*text);
    post("size:%d",size);
    post("file:%s,location:%d",filename,vol);

    return 1;
}

void ex2_eddbclick(t_ex2 *x)
{
    x->ed = ed_new((t_object*)x);
}

```

```
}
```

```
void ex2_edclose (t_ex2 *obj, char **text, long size)
```

```
{
    int i=0,j=0;
    char tempbuffer[256];
    post("window closed");
    post("size:%d",size);

    for(i = 0;i<size;i++)
    {
        if(i == size-1)
        {
            tempbuffer[j] = *((*text)+i);
            tempbuffer[j+1] = '\0';
            post("%s",tempbuffer);
            break;
        }
        else if(13 == *((*text)+i))
        {
            tempbuffer[j] = '\0';
            post("%s",tempbuffer);
            sprintf(tempbuffer, " "); //clear
            j = 0;
        }
        else
        {
            tempbuffer[j] = *((*text)+i);
            j++;
        }
    }
}
```

```
void *ex2_new(void)
```

```
{
    t_ex2 *x;
    char* def = "write something";
    x = (t_ex2*)newobject(ex2_class);
    x->ed = ed_new((t_object*)x);
    ed_vis(x->ed);
    ed_settext(x->ed,&def,15);

    return x;
}
```

ex3.c

```
#include "ext.h"
```

```
// a client
```

```
typedef struct ex3 {  
    t_object b_ob;  
    struct ex3 *c_next;  
    t_symbol *name;  
    void* server;  
    void *out;  
} t_ex3;
```

```
void *ex3_new(Symbol* name);  
void ex3_newserver (t_ex3 *x, void *server);  
void ex3_freeserver (t_ex3 *x, void *server);  
void ex3_listentome (t_ex3 *x, t_symbol *mess, void* value);  
void ex3_free(t_ex3* x);
```

```
void *ex3_class;
```

```
void main(void)  
{  
    setup((t_messlist **)&ex3_class, (method)ex3_new, 0L, (short)sizeof(t_ex3),  
0L, A_SYM,0L);  
    address((method)ex3_newserver, "newserver", A_CANT,0);  
    address((method)ex3_freeserver, "freeserver", A_CANT,0);  
    address((method)ex3_listentome, "listentome", A_LONG,0);  
}
```

```
void *ex3_traverse(t_ex3 *x, t_ex3 ***addr)  
{  
    *addr = &x->c_next;  
    post("traverse");  
    return x->c_next;  
}
```

```
void ex3_newserver (t_ex3 *x, void *server)  
{  
    post("new server created");  
    x->server = server;  
}
```

```
void ex3_freeserver (t_ex3 *x, void *server)  
{  
    post("server freed");  
}
```

```
void ex3_listentome (t_ex3 *x, t_symbol *mess, void* value)  
{  
    long val = *((long*)value);
```

```

        post("received %d from server", val);
        outlet_int(x->out, val);
    }

void *ex3_new(Symbol* name)
{
    t_ex3 *x;
    x = (t_ex3*)newobject(ex3_class);
    x->name = name;
    connection_client((void*)x, name, gensym("ex4"), (method)ex3_traverse);
    x->out = intout(x);

    return x;
}

void ex3_free(t_ex3* x)
{
    connection_delete ((void*)x, x->name);
}

```

ex4.c

```

#include "ext.h"

// a server

typedef struct ex4 {
    t_object b_ob;
    struct ex4 *c_next;
    t_symbol* name;
    long num;
    void *out;
} t_ex4;

void *ex4_new(Symbol* name);
void ex4_free(t_ex4* x);

void ex4_int(t_ex4* x, long input);
void *ex4_class;

void main(void)
{
    setup((t_messlist **)&ex4_class, (method)ex4_new, (method)ex4_free,
    (short)sizeof(t_ex4), 0L, A_SYM, 0L);
    addint((method)ex4_int);
}

void ex4_int(t_ex4* x, long input)
{

```

```

        x->num = input;
    connection_send((void*)x,x->name,gensym("listentome"),(void*)&x->num);
}

void *ex4_new(Symbol* name)
{
    t_ex4 *x;
    x = (t_ex4*)newobject(ex4_class);
    x->name = name;
    connection_server ((void*)x, name);

    return x;
}

void ex4_free(t_ex4* x)
{
    connection_delete ((void*)x, x->name);
}

```

ex5.c

```

#include "ext.h"

// get event serial number

typedef struct {
    t_object b_ob;
} t_ex5;

void ex5_bang(t_ex5 *x);
void *ex5_new(void );

void *ex5_class;

void main(void)
{
    setup((t_messlist **)&ex5_class, (method)ex5_new, 0L, (short)sizeof(t_ex5),
    0L,0L);
    addbang((method)ex5_bang);
}

void ex5_bang(t_ex5 *x)
{
    post("Event Serial Number: %d",evnum_get());
}

void *ex5_new(void )
{
    t_ex5 *x;

```

```
    x = (t_ex5*)newobject(ex5_class);
    return x;
}
```

ex6.c

```
#include "ext.h"
```

```
// check if the event is simultaneous or not
```

```
typedef struct {
    t_object b_ob;
    void *m_proxy;
    long m_inletNumber;
    long lastevent;
} t_ex6;
```

```
void ex6_bang(t_ex6 *x);
void *ex6_new(void );
```

```
void *ex6_class;
```

```
void main(void)
{
    setup((t_messlist **)&ex6_class, (method)ex6_new, 0L, (short)sizeof(t_ex6),
    0L,0L);
    addbang((method)ex6_bang);
}
```

```
void ex6_bang(t_ex6 *x)
{
    int tmp = evnum_get();
    if(x->lastevent == tmp)
    {
        post("simultaneous!");
    }
    x->lastevent = tmp;
}
```

```
void *ex6_new(void )
{
    t_ex6 *x;
    x = (t_ex6*)newobject(ex6_class);
    x->m_proxy = proxy_new(x,1,&x->m_inletNumber);
    return x;
}
```

```
void ex6_free(t_ex6 *x)
{

```



```
    if(x->m_proxy)
        freeobject(x->m_proxy);
}
```

ex7.c

```
#include "ext.h"
```

```
// internal expr
```

```
typedef struct {
    t_object b_ob;
    void *expr;
} t_ex7;
```

```
void ex7_float(t_ex7 *x, float input);
void *ex7_new(void );
```

```
void *ex7_class;
```

```
void main(void)
{
    setup((t_messlist **)&ex7_class, (method)ex7_new,  0L, (short)sizeof(t_ex7),
    0L,0L);
    addfloat((method)ex7_float);
}
```

```
void ex7_float(t_ex7 *x, float input)
{
    t_atom result;
    t_atom args;
    SETFLOAT(&args,input);
    expr_eval(x->expr,1,&args,&result);
    post("%f",result.a_w.w_float);
}
```

```
void *ex7_new(void )
{
    t_ex7 *x;
    t_atom types[10];
    t_atom argv[10];

    x = (t_ex7*)newobject(ex7_class);
    SETSYM(&argv[0],gensym("$f1"));
    SETSYM(&argv[1],gensym("*"));
    SETFLOAT(&argv[2],3.14);
    x->expr = expr_new(3,argv,types);

    return x;
}
```

```

}

void ex7_free(t_ex7 *x)
{
    if(x->expr)
        freeobject(x->expr);
}

```

ex8.c

```

#include "ext.h"

// using presets

typedef struct {
    t_object b_ob;
    int cr;
    void* out;
} t_ex8;

void ex8_bang(t_ex8 *x);
void *ex8_new(void);
void ex8_int(t_ex8 *x, long input);
void ex8_preset(t_ex8 *x);
void *ex8_class;

void main(void)
{
    setup((t_messlist **)&ex8_class, (method)ex8_new, 0L, (short)sizeof(t_ex8),
    0L, 0L);
    address((method)ex8_preset, "preset", 0L);
    addint((method)ex8_int);
    addbang((method)ex8_bang);
}

void ex8_bang(t_ex8 *x)
{
    outlet_int(x->out, x->cr);
}

void ex8_int(t_ex8 *x, long input)
{
    x->cr = input;
}

void ex8_preset(t_ex8 *x)
{

```

```
    preset_int(x,x->cr);  
}  
  
void *ex8_new(void)  
{  
    t_ex8 *x;  
    x = (t_ex8*)newobject(ex8_class);  
    x->cr = 0;  
    x->out = intout(x);  
    return x;  
}
```