C PROGRAMMING

2024

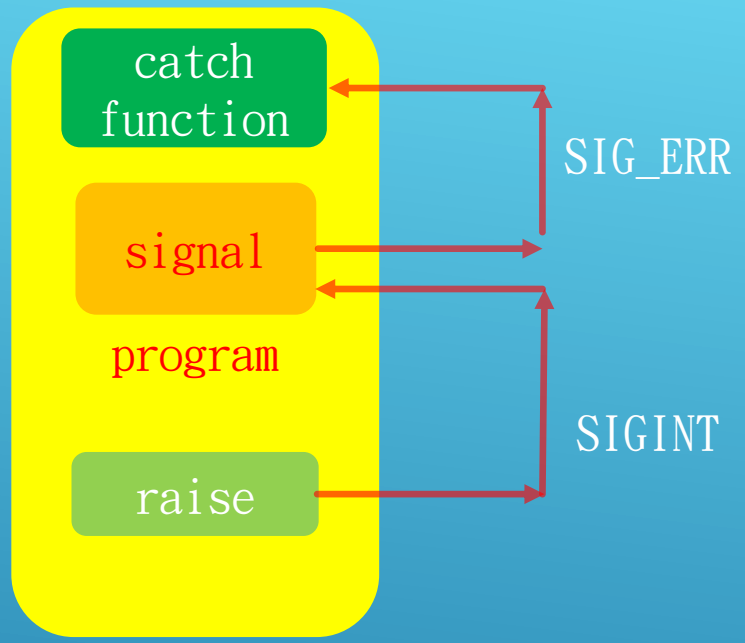
EUGENE KO

STATIC VAIABLES

```
#include <stdio.h>

static int j = 0;    // global variable j declared static

void up(void)
{
          static int k = 0;    // local variable k declared static
          j++;                 // global variable j declared static
          k++;
          printf("up() called. k= %2d, j= %2d\n", k , j);
}
void down(void)
          {
          static int k = 0;    // local variable k declared static
          j--;                 // global variable j declared static
          k--;
          printf("down() called. k= %2d, j= %2d\n", k , j);
}
int main(void){
          int i;
          /* call the up function 3 times, then the down function 2 times */
          for (i = 0; i < 3; i++)        up();
          for (i = 0; i < 2; i++)        down();

          return 0;
}
```

SIGNAL

```c
#include <stdio.h>
#include <stdlib.h>
static void catch_function(int signal) {
        puts("Interactive attention signal caught.");
}
int main(void) {
        if (signal(SIGINT, catch_function) == SIG_ERR) {
                // SIG_ERR is not a valid signal number.
                // When the user presses Ctrl+C,
                // the signalHandler (catch_function) function will be called
                fputs("The Ctrl+C occurred with error under setting a signal handler.\n", stderr);
                return EXIT_FAILURE;
        }
        puts("Raising the interactive attention signal.");
        if (raise(SIGINT) != 0) {
        // the raise function is used to send a signal to the current process.
        // It is the interrupt signal, often generated when the user presses Ctrl+C in the terminal
                fputs("Error raising the signal.\n", stderr);
                return EXIT_FAILURE;
        }
        puts("Exiting.");
        return 0;
}
```



SETJMP

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

// Declare a global variable to hold the jump buffer
jmp_buf jump_buffer;

void functionWithJump() {
    printf("Inside functionWithJump\n");
    // Setjmp returns 0 when called directly, but returns a non-zero value
    // when returning from longjmp
    if (setjmp(jump_buffer) != 0) {
        // This code is executed when longjmp is called
        printf("Returning from longjmp\n");
        return;
    }
    printf("About to call longjmp\n");

    // Use longjmp to jump back to the point of setjmp
    longjmp(jump_buffer, 1);
}
int main() {
    printf("Before calling functionWithJump\n");

    // Call the function with setjmp
    functionWithJump();

    printf("After calling functionWithJump\n");

    return 0;
}
```