

stdlib.h



- □□□□□□
- abs() labs() llabs()
- div() ldiv() lldiv()
- □□□□□□
 - a □□□□
 - str □□□□□□□□□□
 - str □□□□□□□□□□
- rand()
- srand()
- abort()
- exit() quick_exit() _Exit()
- atexit() at_quick_exit()
- getenv()
- system()
- □□□□□□
 - aligned_alloc()
- qsort()
- bsearch()
- □□□□□□



stdlib.h □□□□□□□□□□

- `size_t` `sizeof` `□□□□□`
- `wchar_t` `□□□□□□`

stdlib.h

--	--	--	--	--	--	--	--

- NULL
- EXIT_SUCCESS
- EXIT_FAILURE
- RAND_MAX rand()
- MB_CUR_MAX

abs() labs() llabs()

abs() int labs() long int llabs() long long int

```
int abs(int j);
long int labs(long int j);
long long int llabs(long long int j);
```

```
// -2 的绝对值 = 2
printf("-2 的绝对值 = %d\n", abs(-2));

// 4 的绝对值 = 4
printf("4 的绝对值 = %d\n", abs(4));
```

div() ldiv() lldiv()

div() int ldiv() long int lldiv() long long int

```
div_t div(int numer, int denom);
ldiv_t ldiv(long int numer, long int denom);
lldiv_t lldiv(long long int numer, long long int denom);
```

div() div_t ldiv() ldiv_t lldiv() lldiv_t

```
int quot; //商
int rem; //余数
```

□□□□□□□□

```
typedef struct {
    int quot, rem;
} div_t;

typedef struct {
    long int quot, rem;
} ldiv_t;

typedef struct {
    long long int quot, rem;
} lldiv_t;
```

□□□□□□□

```
div_t d = div(64, -7);

// □□ 64 / -7 = -9
printf("64 / -7 = %d\n", d.quot);

// □□ 64 % -7 = 1
printf("64 %% -7 = %d\n", d.rem);
```

□□□□□□□□

a □□□□

stdlib.h □□□□□□□□□□□□□□□□

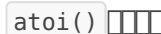
- atoi()□□□□□ int □□□
- atof()□□□□□ double □□
- atol()□□□□□ long int □□
- atoll()□□□□□ long long int □□

□□□□□□□

```
int atoi(const char* nptr);
double atof(const char* nptr);
```

```
long int atol(const char* nptr);
long long int atoll(const char* nptr);
```

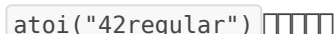


  a  ASCII  atoi()  "ASCII to int"


 0



```
atoi("3490")    // 3490
atof("3.141593") // 3.141593
```


 atoi()  atoi("42regular")  42

 "hello world"  0

str

stdlib.h 

- strtod()  float 
- strtod()  double 
- strtold()  long double 



```
float strtod(
    const char* restrict nptr,
    char** restrict endptr
);
```

```
double strtod(
    const char* restrict nptr,
    char** restrict endptr
);
```

```
long double strtold(
    const char* restrict nptr,
    char** restrict endptr
);
```

[illegible]

- nptr 
 - endprt 
-   \0  NULL 

ERANGE

```
char *inp = "    123.4567abdc";
char *badchar;

double val = strtod(inp, &badchar);

printf("%f\n", val); // 123.456700
printf("%s\n", badchar); // abdc
```

```
if (*endptr == '\\0') {
    // 空白
} else {
    // 文字列
}
```

INF $\square$ INFINITY

NAN□□□□

`str`

- `strtol()`  `long int` 
- `strtoll()`  `long long int` 
- `strtoul()`  `unsigned long int` 
- `strtoull()`  `unsigned long long int` 

--	--	--	--	--	--	--	--	--

2 endPrt

☐ \0

2 36

0

--	--	--	--	--	--	--	--

[illegible]

```
char* s = "3490";
unsigned long int x = strtoul(u, NULL, 10);
```

```
printf("%lu\n", x); // 3490
```

```
char* end;

long value = strtol("0xff", &end, 16);
printf("%ld\n", value); // 255
printf("%s\n", end); //  



value = strtol("0xffxx", &end, 16);
printf("%ld\n", value); // 255
printf("%s\n", end); // xx
```

```
char* s = "101010";  
unsigned long int x = strtoul(s, NULL, 2);  
  
printf("%lu\n", x); // 42
```

```
#include <stdio.h>

#include <stdlib.h>

int main(void) {
    const char* string = "-1234567abc";
    char* remainderPtr;
    long x = strtol(string, &remainderPtr, 0);
    printf("%s\\\"%s\\\"\\n%s%ld\\n%s\\\"%s\\\"\\n",
        "The original string is ",
        string,
        "The converted value is ",
        x,
        "The remainder of the original string is ",
        remainderPtr
    );
}
```

□□□□□□□□

The original string is "-1234567abc"
The converted value is -1234567
The remainder of the original string is "abc"

□□□□□□□□

strtol() □□□

errno □□□

ERANGE □□□□□□

LONG_MIN □□□□□□□□

LONG_MAX □□□□□□□□

strtoul() □□□

ULONG_MAX □

rand()

rand() □□□□□□

0 □ RAND_MAX □□□□□□□□

RAND_MAX □□□□□□

stdlib.h

□□□□□□□□

INT_MAX□

```
// □□  
int rand(void);  
  
// □□  
int x = rand();
```

□□□□□□□□

N □ M □□□□□□□□

N □ M □□□□□□□□□□□□

```
int x = rand() % □M - N + 1) + N;
```

□□□ 1 □ 6 □□□□□□□□□□

```
int x = rand() % 6 + 1;
```

□□□□□□□□□□□□□□□□

```
// 0 □ 0.999999 □□□□□□  
printf("0 to 0.99999: %f\n", rand() / ((float)RAND_MAX + 1));  
  
// n □ m □□□□□□□□  
// n + m * (rand() / (float)RAND_MAX)  
printf("10.5 to 15.7: %f\n", 10.5 + 5.2 * rand() / (float)RAND_MAX);
```

□□□□□□□□

rand() □ RAND_MAX □□ int □□□□□□□□□□□□□□

srand()

rand()  srand() 

srand()  unsigned int 

```
void srand(unsigned int seed);
```

time(NULL) srand()

```
#include <time.h>

srand((unsigned int) time(NULL));
```

The diagram illustrates the memory layout of the time module. It shows several variables and their sizes in bytes:

- `time()`: 16 bytes
- `time.h`: 16 bytes
- `time_t`: 8 bytes
- `time()`: 16 bytes
- `time_t`: 8 bytes
- `NULL`: 8 bytes
- `NULL`: 4 bytes
- `0`: 8 bytes
- `time(0)`: 4 bytes

abort()

abort()  SIGABRT

```
void abort(void);
```



exit() quick_exit() _Exit()

```
void exit(int status);  
void quick_exit(int status);  
void _Exit(int status);
```

The diagram shows a 16-bit register represented as a horizontal bar divided into three sections. The first section on the left is labeled 'EXIT_FAILURE' and contains 10 empty boxes. The middle section contains a single box with the value '0'. The third section on the right is labeled 'EXIT_SUCCESS' and contains 5 empty boxes.

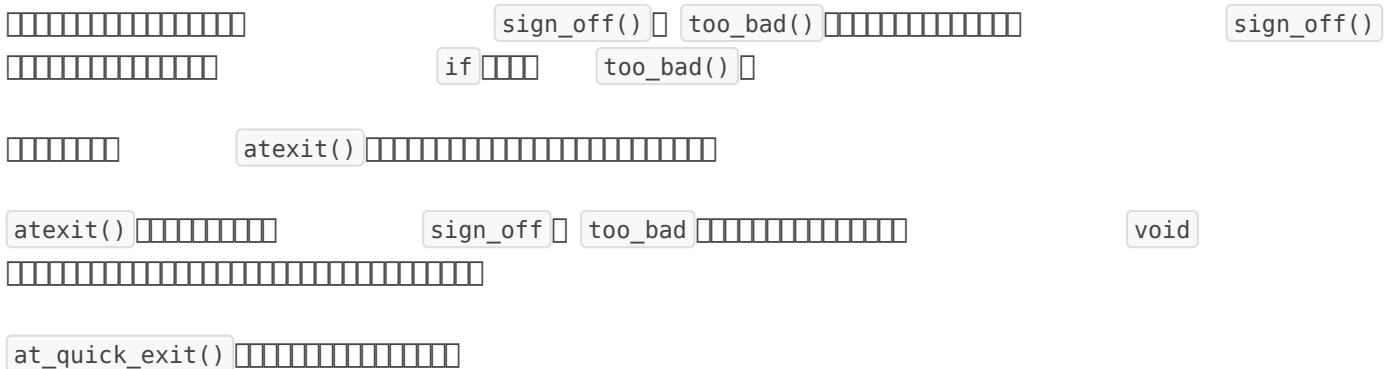

```

printf("%d is %s.\n", n, (n % 2 == 0) ? "even" : "odd");
return 0;
}

void sign_off(void) {
    puts("sign_off");
}

void too_bad(void) {
    puts("too bad");
}

```



```

void exit_handler_1(void) {
    printf("1\n");
}

void exit_handler_2(void) {
    printf("2\n");
}

int main(void) {
    at_quick_exit(exit_handler_1);
    at_quick_exit(exit_handler_2);
    quick_exit(0);
}

```

1 2

getenv()

getenv() 

```
char* getenv(const char* name);
```



 NULL 

 \$PATH 

```
printf("PATH is %s\n", getenv("PATH"));
```

system()

system() 

```
void system( char const * command );
```

 NULL



 ls 

```
system("ls -l");
```



stdlib.h 

- malloc() 
- calloc() 
- realloc() 
- free() 

aligned_alloc()







aligned_alloc()

```
void* aligned_alloc(size_t alignment, size_t size);
```

- alignment  2  2 4 8 16.....
- size 

NULL

```
char* p = aligned_alloc(64, 256);
```

--	--	--	--	--	--

aligned_alloc()

[illegible]256 ☐ ☐ ☐

qsort()

qsort()

```
void qsort(  
    void *base,  
    size_t nmemb,  
    size_t size,  
    int (*compar)(const void *, const void *)  
);
```

- base 
- nmemb 
- size 
- compar 

--	--	--	--

compar

[illegible]

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

```
int compar(const void* elem0, const void* elem1) {
```

```
const int* x = elem0;

const int* y = elem1;


return *x - *y;

}


int main(void) {

    int a[9] = {14, 2, 3, 17, 10, 8, 6, 1, 13};


    qsort(a, 9, sizeof(int), compar);


    for (int i = 0; i < 9; i++)
        printf("%d ", a[i]);

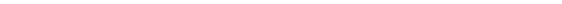
    putchar('\n');

}
```

"1 2 3 6 8 10 13 14 17"□

bsearch()

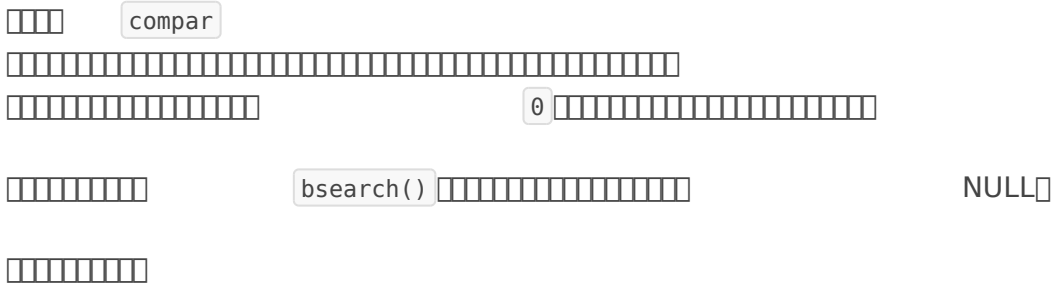
bsearch()

[illegible]

```
void *bsearch(
    const void* key,
    const void* base,
    size_t nmemb,
    size_t size,
    int (*compar)(const void *, const void *))
);
```

 5

- key[10]
- base[20]
- nmemb[8]
- size[10]
- compar[20]



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

int compar(const void *key, const void *value) {
    const int* k = key;
    const int* v = value;

    return *k - *v;
}

int main(void) {
    int a[9] = {2, 6, 9, 12, 13, 18, 20, 32, 47};

    int* r;
    int key;

    key = 12; // 12
    r = bsearch(&key, a, 9, sizeof(int), compar);
    printf("Found %d\n", *r);

    key = 30; // 30
    r = bsearch(&key, a, 9, sizeof(int), compar);
    if (r == NULL)
        printf("Didn't find 30\n");

    return 0;
}
```

Found 12

Didn't find 30



stdlib.h

- mblen()
- mbtowc()
- wctomb()
- mbstowcs()
- wcstombs()

	#2	
		2021-09-04 22:34:34 UTC
		2022-02-17 17:22:23 UTC