



시스템프로그래밍기초 실습 6주차

strtok (string.h), sprintf (stdio.h)

char *strtok(char *str, const char *delim)

- str : The contents of this string are modified and broken into smaller strings (tokens).
- delim : This is the C string containing the delimiters. These may vary from one call to another.

int sprintf(char *str, const char *format, ...)

- str : This is the pointer to an array of char elements where the resulting C string is stored.
- format : This is the String that contains the text to be written to buffer.

실습 예제 1) strtok.c

gument.c

C strtok.c ×

C fib.c

C hanoi.c

C maze.c

C

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main()
5  {
6      char str[100] = { 0, };
7
8      int i, len = 0;
9      for (i = 1; i < 35; i++) {
10         len += sprintf(str + len, "%d", i);
11     }
12     printf("str: %s\n", str);
13
14     char *ptr = strtok(str, "0");
15     while (ptr != NULL) {
16         printf("%s\n", ptr);
17         ptr = strtok(NULL, "0");
18     }
19
20     return 0;
21 }
```

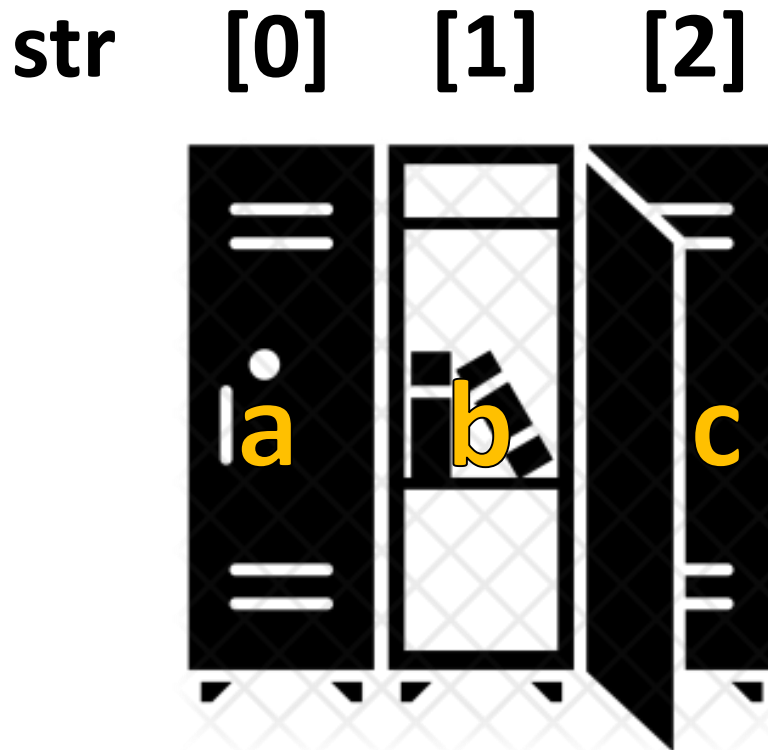
실습 예제 1) strtok.c 결과

```
spubuntu@sp:~/sysPro/spWeek6$ gcc -o my_strtok my_strtok.c
spubuntu@sp:~/sysPro/spWeek6$ ./strtok
str: 12345678910111213141516171819202122232425262728293031323334
1234567891
1112131415161718192
2122232425262728293
31323334
```

int main(int argc, char *argv[])

- int argc : 메인 함수에 전달되는 인자의 개수
- char *argv[] : 프로그램을 실행할 때 지정해 준 문자열들이 실제로 저장되는 배열
 - argv[0] : 실행 파일명
 - argv[1] : 띄어쓰기 후 입력된 문자열
 - ○
○
 - argv[n] : 띄어쓰기 후 입력된 문자열

pointer



- CODE

```
test.c  x
1  #include <stdio.h>
2
3  int main(void) {
4      char *str = "abc";
5
6      if (*str=='a'){
7          printf("* is right\n");
8      }
9      printf("str[1] address: %p\n", &str[1]);
10     if (*&str[1]=='b'){
11         printf("char %c\n",str[1]);
12     }
13     return 0;
14 }
```

- RESULT

```
51323334
spubuntu@sp:~/sysPro/spWeek6$ ./test
* is right
str[1] address: 0x400645
char b
```

실습 예제 2) argument.c

```
C argument.c x C maze.c
1  #include <stdio.h>
2  #include <stdlib.h> // atoi
3  #include <ctype.h>  // isdigit
4
5  int main(int argc, char *argv[])
6  {
7      int i;
8      int sum = 0;
9
10     if (argc < 3) {
11         printf("Too few argument(less than 3)\n");
12         return 1;
13     }
14     for (i=0; i<argc; i++) {
15         printf("argv[%d]: %s\n", i, argv[i]);
16         if (isdigit(*argv[i])) {
17             sum += atoi(argv[i]);
18         }
19     }
20     printf("Sum of argument: %d\n", sum);
21     return 0;
22 }
```

실습 예제 2) argument.c 결과

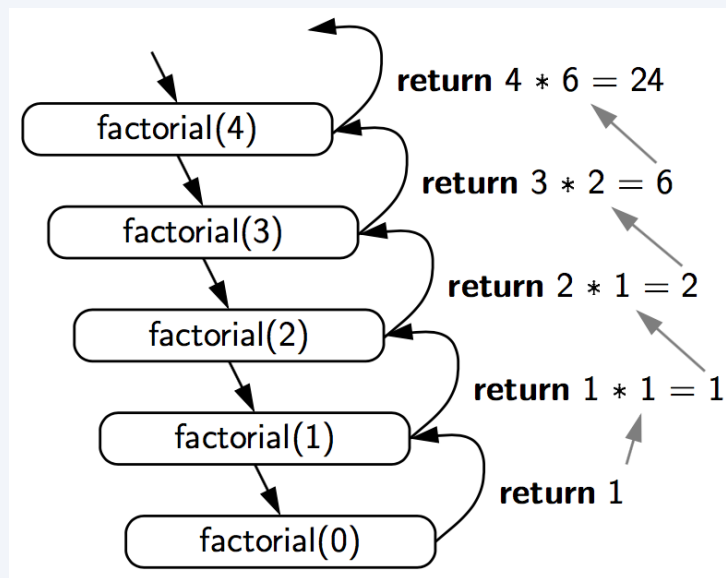
```
spubuntu@sp:~/sysPro/spWeek6$ ./argument 11
Too few argument(less than 3)
spubuntu@sp:~/sysPro/spWeek6$ ./argument 11 23 16 9
argv[0]: ./argument
argv[1]: 11
argv[2]: 23
argv[3]: 16
argv[4]: 9
Sum of argument: 59
spubuntu@sp:~/sysPro/spWeek6$ ./argument 11 23
argv[0]: ./argument
argv[1]: 11
argv[2]: 23
Sum of argument: 34
spubuntu@sp:~/sysPro/spWeek6$
```


재귀 함수

- 자기 함수 내에서 자기 자신을 호출하는 함수.

- 대표적인 예:

- factorial 함수
- 피보나치 수열
- 하노이의 탑
- 미로 찾기



실습 예제 3) fib.c

```
fib.c  x
1  #include <stdio.h>
2
3  int fib(int);
4
5  int main()
6  {
7      int n, i;
8      printf("Type fibonacci number: ");
9      scanf("%d", &n);
10
11     if (n < 1) {
12         printf("Integer n must be 1 at least.\n");
13         return 1;
14     }
15     /* 1, 1, 2, 3, 5, 8, ... */
16     printf("%dth fibonacci number is %d\n", n, fib(n));
17
18     return 0;
19 }
20
21 int fib(int num)
22 {
23     /* base case */
24     if (num == 1 || num == 2)
25         return 1;
26     /* case when num > 1 */
27     else
28         return fib(num - 1) + fib(num - 2);
29 }
```

실습 예제 3) fib.c 결과

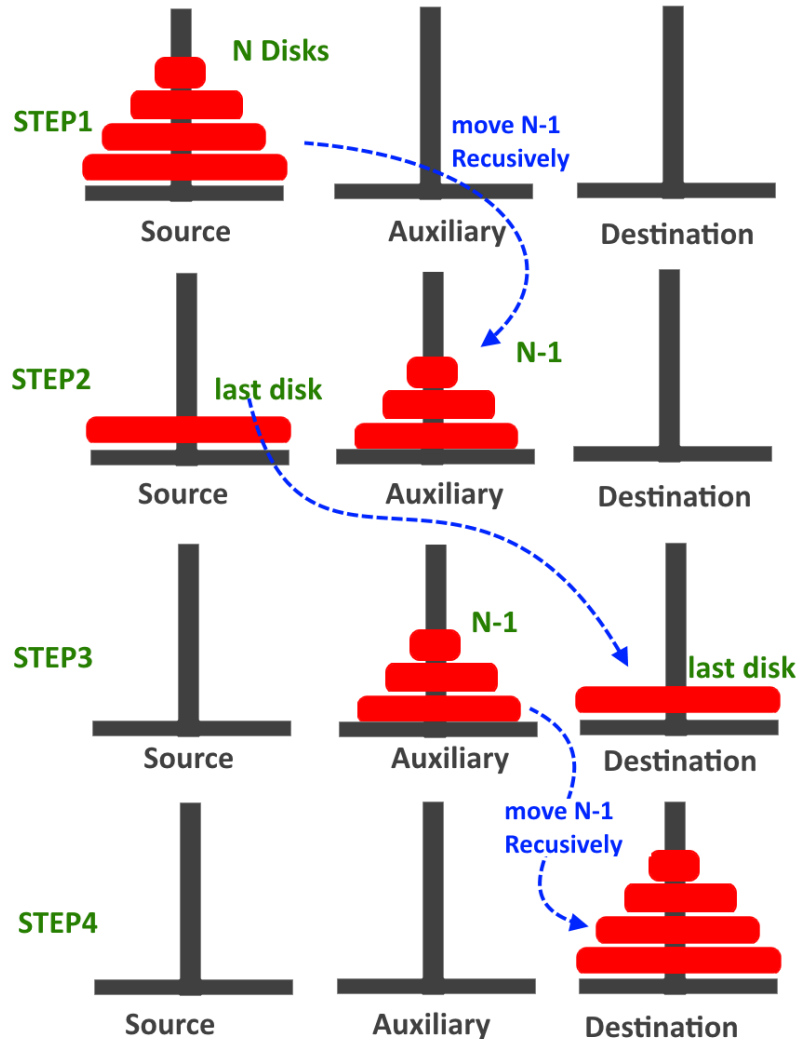
```
spubuntu@sp:~/sysPro/spWeek6$ ./fib
Type fibonacci number: 3
3th fibonacci number is 2
spubuntu@sp:~/sysPro/spWeek6$ ./fib
Type fibonacci number: 9
9th fibonacci number is 34
spubuntu@sp:~/sysPro/spWeek6$ ./fib
Type fibonacci number: 10
10th fibonacci number is 55
spubuntu@sp:~/sysPro/spWeek6$ ./fib
Type fibonacci number: 1
1th fibonacci number is 1
spubuntu@sp:~/sysPro/spWeek6$
```

assert (assert.h)

```
void assert(int expression);
```

- Abort the program if expression is false.
- 수식의 값이 기대하고 있는 값인가를 확인할 때 사용.
- NDEBUG가 정의되어 있으면, 모든 assert는 무시됨.

실습 예제 3) hanoi.c 개념



== PRINCIPLE ==

N 개의 원판을 Source 기둥에서 Destination 기둥으로 옮긴다.

(단, 한 번에 하나의 원판만 옮길 수 있으며, 큰 원판이 작은 원판 위에 있어서는 안 된다.)

== STEP ==

1. 가장 큰 원판을 제외한 N-1 개의 원판을 Auxiliary 기둥으로 옮긴다.
2. 가장 큰 원판을 Destination 기둥으로 옮긴다.
3. step1-2를 반복하여, 1에서 옮긴 N-1 개의 원판을 다시 Destination 기둥으로 옮긴다.

실습 예제 4) hanoi.c (1)

```
C test.c  C hanoi.c  x
1  #include <stdio.h>
2  #include <assert.h>
3  #include <stdlib.h>
4
5  int cnt = 0;
6
7  void move(int, char, char, char);
8  int get_n_from_user(void);
9  |
10 int main()
11 {
12     int n;
13     n = get_n_from_user();
14     assert(n > 0 && n < 7);
15     /*
16     // Move n disks from tower A tower A to tower C,
17     // using tower B as an intermediate tower.
18     */
19     move(n, 'A', 'B', 'C'); /* recursive function */
20
21     return 0;
22 }
```

실습 예제 4) hanoi.c (2)

```
24 void move(int n, char a, char b, char c)
25 {
26     if (n == 1) {
27         cnt++;
28         printf("%5d: %s%d%s%c%s%c. \n", cnt,
29             "Move disk ", 1, " from tower ", a, " to tower ", c);
30     }
31     else {
32         move(n - 1, a, c, b);
33         cnt++;
34         printf("%5d: %s%d%s%c%s%c. \n", cnt,
35             "Move disk ", n, " from tower ", a, " to tower ", c);
36         move(n - 1, b, a, c);
37     }
38 }
39
40 int get_n_from_user(void)
41 {
42     int n;
43     printf("%s",
44         "---\n"
45         "TOWER OF HANOI:\n"
46         "\n"
47         "Input n: ");
48     if (scanf("%d", &n) != 1 || n < 1) {
49         printf("\nERROR: Positive integer not found - bye!\n\n");
50         exit(1);
51     }
52     printf("\n");
53
54     return n;
55 }
56
```

실습 예제 4) hanoi.c 결과

```
spubuntu@sp:~/sysPro/spWeek6$ ./hanoi
---
TOWER OF HANOI:

Input n: 7

hanoi: hanoi.c:14: main: Assertion `n > 0 && n < 7' failed.
Aborted (core dumped)
```

```
spubuntu@sp:~/sysPro/spWeek6$ ./hanoi
---
TOWER OF HANOI:

Input n: 4

1: Move disk 1 from tower A to tower B.
2: Move disk 2 from tower A to tower C.
3: Move disk 1 from tower B to tower C.
4: Move disk 3 from tower A to tower B.
5: Move disk 1 from tower C to tower A.
6: Move disk 2 from tower C to tower B.
7: Move disk 1 from tower A to tower B.
8: Move disk 4 from tower A to tower C.
9: Move disk 1 from tower B to tower C.
10: Move disk 2 from tower B to tower A.
11: Move disk 1 from tower C to tower A.
12: Move disk 3 from tower B to tower C.
13: Move disk 1 from tower A to tower B.
14: Move disk 2 from tower A to tower C.
15: Move disk 1 from tower B to tower C.
```


2차원배열

```
/* inside the c code. */
```

```
int matrix[3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

```
int matrix2[3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};
```

| | | | |
|--------|--------|--------|--------|
| [0][0] | [0][1] | [0][2] | [0][3] |
| [1][0] | [1][1] | [1][2] | [1][3] |
| [2][0] | [2][1] | [2][2] | [2][3] |

실습 과제 1) maze.c (1)

```
C fib.c  C maze_ex.c x
1  #include <stdio.h>
2  #define      N      8
3
4  const int UNVISITIED_WAY = 0; // 아직 한 번도 가보지 못한 cell
5  const int WALL = 1;         // 벽이라고 정해진 cell
6  const int BLOCKED = 3;      // visited이며 출구까지의 경로가 있지 않음이 밝혀진 cell
7  const int POSSIBLE_WAY = 2; // visited이며 아직 출구로 가는 경로가 될 가능성이 있는 cell
8  int maze[N][N] = {
9      {0, 0, 0, 0, 0, 0, 0, 1},
10     {0, 1, 1, 0, 1, 1, 0, 1},
11     {0, 0, 0, 1, 0, 0, 0, 1},
12     {0, 1, 0, 0, 1, 1, 0, 0},
13     {0, 1, 1, 1, 0, 0, 1, 1},
14     {0, 1, 0, 0, 0, 1, 0, 1},
15     {0, 0, 0, 1, 0, 0, 0, 1},
16     {0, 1, 1, 1, 0, 1, 0, 0}
17 };
18
19 int findPath(int x, int y) {
20     // TO BE IMPLEMENTED
21     // YOUR CODE HERE
22     //
23     // Set the current position value of maze to 2 if possible way exists.
24     // If there is no way, set the current position value of maze to 3.
25     //
26     // hint 1. consider the wall
27     // hint 2. consider the edge
28     // hint 3. should not go to visited place
29 }
30
```

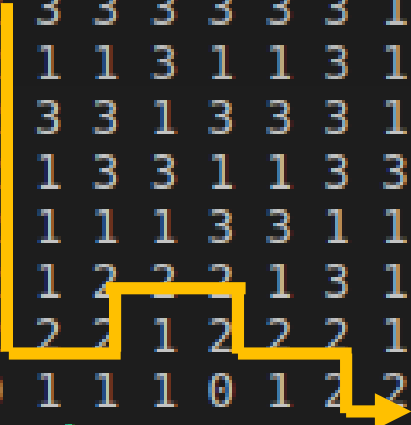


실습 과제 1) maze.c (2)

```
30
31  int main() {
32      int i, j;
33      printf("Escape the maze from (0,0) to (%d,%d) : %d\n", N-1, N-1, findPath(0, 0));
34      for (i = 0; i < N; i++) {
35          for (j = 0; j < N; j++) {
36              printf("%d ", maze[i][j]);
37          }
38          printf("\n");
39      }
40      return 0;
41  }
```

실습 과제 1) maze.c 결과

```
15: Move disk 1 from tower B to tower C.  
spubuntu@sp:~/sysPro/spWeek6$ ./maze  
Escape the maze from (0,0) to (7,7) : 1  
2 3 3 3 3 3 3 1  
2 1 1 3 1 1 3 1  
2 3 3 1 3 3 3 1  
2 1 3 3 1 1 3 3  
2 1 1 1 3 3 1 1  
2 1 2 2 2 1 3 1  
2 2 2 1 2 2 2 1  
0 1 1 1 0 1 2 2
```



실습 과제 2) my_strtok.c

```
1  #include <stdio.h>
2
3  int main()
4  {
5      char str[100] = { 0, };
6      int i, len = 0;
7      for (i = 1; i < 35; i++) {
8          len += sprintf(str + len, "%d", i);
9      }
10
11     // TO BE IMPLEMENTED
12     // YOUR CODE HERE
13     // Split the given string with '0'.
14
15     return 0;
16 }
```



실습 과제 2) my_strtok.c 결과

```
spubuntu@sp:~/sysPro/spWeek6$ ./my_strtok  
1234567891  
1112131415161718192  
2122232425262728293  
31323334
```

과제 제출 방법

1. 모든 파일은 **sys_06_학번.tar.gz**으로 압축하여 제출한다.
2. 메일 제목은 **[시프기]_06_이름_학번**으로 한다.
3. 제출 파일들을 빈 디렉토리에 넣고 그 디렉토리 안으로 이동한 후,
다음과 같이 압축 명령어를 사용한다.(폴더가 아닌 파일들만 압축한다.)

```
$ tar -zcvf sys_06_학번.tar.gz *
```

제출 파일

- | | |
|---------------|-----------------|
| 1. strtok.c | 5. maze.c |
| 2. argument.c | 6. my_strtok.c |
| 3. fib.c | 7. script_week6 |
| 4. hanoi.c | |



감사합니다.
