System Programming
숙제 2

인천대학교 | 컴퓨터공학부

시스템프로그래밍 | 지도교수 박문주

송병준 | 201701562

0.목차

System Programming 숙제 2

CSAPP 2장 문제 풀이

송병준
2019년 9월 22일

---

1. 과제

Computer System, A Programmer's Perspective 2장 연습 문제 풀이
- 2.74
- 2.76
- 2.80

2. 환경

운영체제:	macOS 10.14.6 (Darwin Kernel Version 18.7.0 x86_64)
프로세서:	Intel Core i5
편집기:	vim 8.0.1365
빌드:	Apple LLVM version 10.0.1 (clang-1001.0.46.4)

3. 문제 풀이

3.1. 문제 2.74

다음과 같은 프로토타입을 갖는 함수를 작성하시오.

```
/* Determine whether arguments can be subtracted without overflow */
int tsub_ok(int x, int y);
```

이 함수는 x-y가 오버플로우를 발생시키지 않으면 1을 리턴한다.

뺄셈 연산시 오버플로를 검사하는 함수이다. 오른쪽 피연산자의 부호만 바꾸어서 덧셈 오버플로 검사 함수를 사용하면 된다.
이때 y가 가장 작은 값을 가지는 음수일 경우에는 음수를 취하여도 값에 변화가 없게 된다. 따라서 y가 (0x1<<비트폭-1)인 경우만 조심하면 된다.

[소스코드]

```
#define MSB(x) ((x) & (1 << ((sizeof(typeof(x)) << 3) - 1)))

static inline int tadd_ok(int x, int y) {
	return (MSB(x) != MSB(y) || MSB(x + y) == MSB(x));
}

int tsub_ok(int x, int y) {
	return (y != (1 << 31) && tadd_ok(x, -y));
}
```

[테스트 결과]

```
==== Test 3 Started ====
Assertion succeeded: tsub_ok(1, -1) == 1
Assertion succeeded: tsub_ok(2147483647, -2147483648) == 0
Assertion succeeded: tsub_ok(-2147483647, 2) == 0
Assertion succeeded: tsub_ok(-2147483647, 1) == 1
Assertion succeeded: tsub_ok(-1, 2147483647) == 1
==== Test 3 Succeeded ====
```

3.2. 문제 2.76

라이브러리 함수 calloc은 다음과 같이 선언된다.

```
void *calloc(size_t nmemb, size_t size);
```

라이브러리 문서에는 다음과 같이 설명되어 있다. "calloc함수는 각각 size 바이트 크기를 갖는 nmemb 배열에 메모리를 할당한다. 이 메모리는 0으로 설정된다. 만일 nmemb나 size가 0이면 calloc은 NULL을 리턴한다."

malloc을 호출해서 메모리를 할당하고, memset으로 값을 0으로 설정하도록 calloc 함수를 구현하시오. 여러분의 코드는 산술 오버플로우로 인한 취약성이 있어서는 안 되며, 자료형 size_t를 표시하기 위해 사용되는 비트 수에 관계없이 정확히 동작해야 한다.

malloc과 memset함수는 다음과 같이 정의된다는 점을 참고하시오.

```
void *malloc(size_t size);
void *memset(void *s, int c, size_t n);
```

calloc은 주어진 두 인자를 곱한 크기만큼 메모리를 할당한 뒤 0으로 초기화하여 반환하는 함수이다. 이때 두 비부호형 정수를 곱하는 과정에서 오버플로가 발생할 수 있다.
두 수를 곱할 때에 오버플로가 발생하면 필히 정보의 손실이 발생한다. 즉 역연산의 결과가 원래의 값과 같지 않다는 것이다. 이를 이용하여 오버플로를 검출할 수 있다.

[소스코드]

```
static inline int size_t_mult_ok(size_t x, size_t y) {
	return (x == 0 || (x*y)/x == y);
}

void *calloc_impl(size_t count, size_t size) {
	if (!(count && size)) {
		// Not an error.
		return NULL;
	}

	if (!size_t_mult_ok(count, size)) {
		// Definitely an error.
		errno = ENOMEM;
		return NULL;
	}
```

```
        size_t alloc_size = count * size;
        void *allocated = malloc(alloc_size);

        if (allocated == NULL) {
                // Error in malloc.
                return NULL;
        }

        if (memset(allocated, 0, alloc_size) == NULL) {
                // Error in memset.
                return NULL;
        }

        return allocated;
}
```

[테스트 결과]

```
==== Test 4 Started ====
Assertion succeeded: (allocated = calloc_impl(0, 0)) == NULL
Assertion succeeded: (allocated = calloc_impl(0, 30)) == NULL
Assertion succeeded: (allocated = calloc_impl(30, 0)) == NULL
Assertion succeeded: (allocated = calloc_impl(5, 64)) != NULL
Assertion succeeded: (allocated = calloc_impl(1L<<63, 1L<<62)) == NULL
==== Test 4 Succeeded ====
```

3.2. 문제 2.80

> 정수 인자 x에 대해 $\frac{3}{4}$x를 계산한 후에 0 방향으로 근사하는 함수 threefourths를 작성하라. 이 함수는 오버플로우가 발생하면 안 된다. 작성한 함수는 비트수준 정수 코딩 규칙 (124쪽)을 따라야 한다.

보통 분수를 계산할 때에는 나눗셈으로 인한 손실을 막기 위해 먼저 곱한 뒤 나누는 방식을 택한다. 하지만 해당 방법에서는 오버플로가 발생할 여지가 있다. 곱셈 오버플로가 발생하면 정보의 손실이 발생하기 때문에 바람직하지 않다.
따라서 먼저 나눈 뒤 곱하는 방법으로 접근한다. 이 때의 문제는 나눔을 먼저 함으로 인해 오차가 쌓이는 것인데, 이를 해결하기 위해 나눗셈으로 잘리는 수를 따로 보관하였다가 마지막에 합산하여 준다.

[소스코드]

```
int threefourth(int x) {
        int num = 3;
        int denom = 4;

        int raw_result = ((x / denom) * num);
        int lose = x - ((x / denom) * denom);
        int bias = (lose * num) / denom;

        return raw_result + bias;
}
```

[테스트 결과]

```
==== Test 5 Started ====
Assertion succeeded: threefourth(4) == 3
Assertion succeeded: threefourth(891) == 668
Assertion succeeded: threefourth(2147483647) == 1610612735
Assertion succeeded: threefourth(-4) == -3
Assertion succeeded: threefourth(-891) == -668
Assertion succeeded: threefourth(-2147483647) == -1610612735
Assertion succeeded: threefourth(-2147483648) == -1610612736
==== Test 5 Succeeded ====
```

4. 마무리

연습문제는 [컴퓨터 시스템 3판]에서 발췌했습니다.