

Size_t Ssize_t

2019931050 이승희

int Unsigned int

int

- 최소 16비트 이상, short의 크기 이상인 정수형
Ex) 마인크래프트의 최고 레벨 지역

Unsigned int

- int의 범위를 양을 정수로만 사용하는 정수형
Ex) 바르누의 나라 최대 경험치 42.9억

Size_t SSize_t

Size_t

"The Data Type Size_t is unsigned integral type"

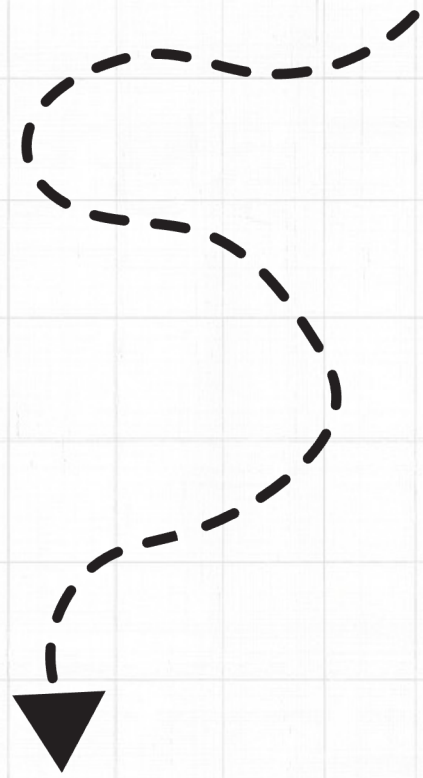
- 이론상 가장 큰 사이즈를 담을 수 있는 unsigned int
32비트 컴퓨터에선 int와 차이가 없을 수도 있다

SSize_t

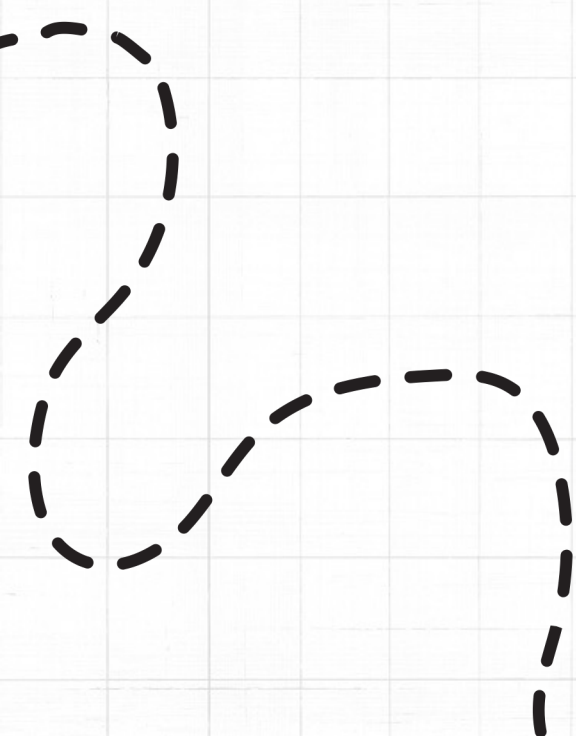
"The Data Type SSize_t is signed integral type"

Size_t

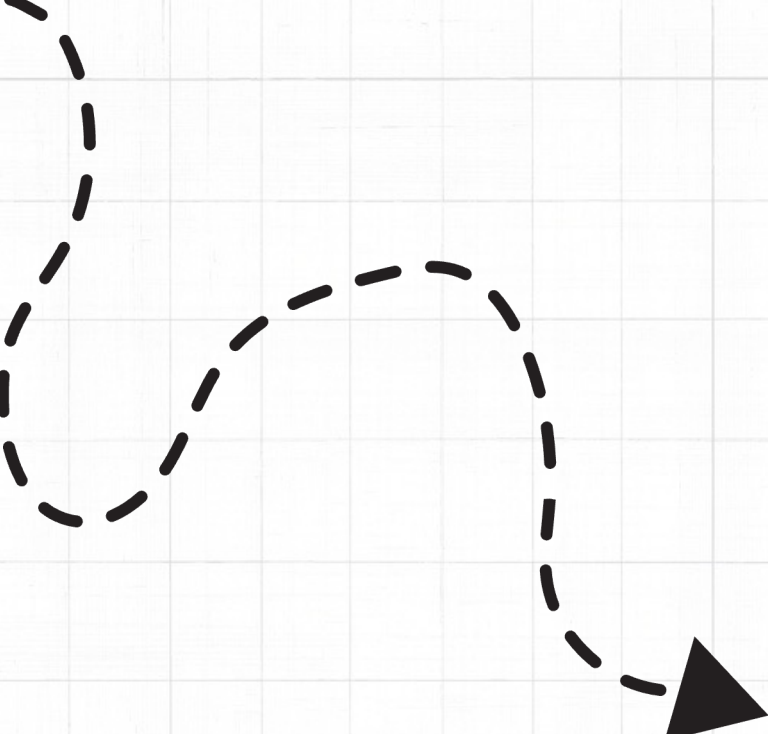
왜 사용하는가?



int로 표현하기에 사이즈가 너무 커지면 오버플로가 나타난다
따라서 int 대신 size_t를 사용하여 오류를 막기 위해서다



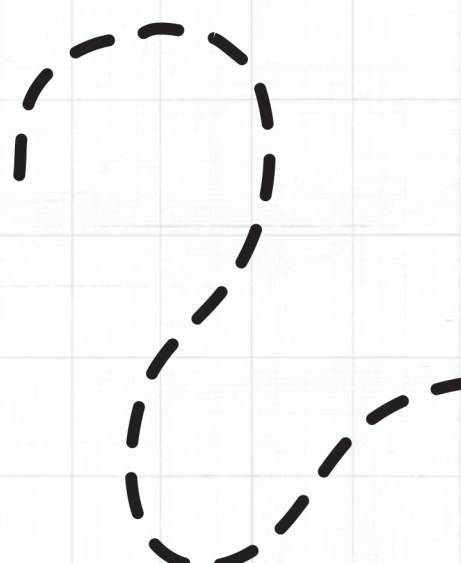
read() 또는 write() 함수의 경우 실패 시 -1을 반환할 수 있기에
부호가 존재하는 ssize_t를 사용함으로 오버플로를 막을 수 있다



Why?

Unsigned int가 아닌 size_t를 사용하는 이유는
size_t 자료형은 해당 시스템에서 포함할 수 있는
최대 크기의 데이터를 정의한다

하지만 Unsigned int는 32bit, 64bit 운영체제마다 무조건
32 bit, 64bit로 정의되지 않는다





```
arch LONG_BIT
```



```
getconf LONG_BIT
```


↑ ↓ ↑ ↓
Σ Π



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

```
int main(void) {  
    printf("size of int : %lu\n", sizeof(int));  
    printf("size of unsigned int : %lu\n", sizeof(unsigned int));  
    printf("size of size_t : %lu\n", sizeof(size_t));  
    printf("size of ssize_t : %lu\n", sizeof(ssize_t));  
}
```



```
gcc -o test1.c test1.c
```

```
size of int : 4  
size of unsigned int : 4  
size of size_t : 8  
size of ssize_t : 8
```

↑
Σ
↑
10



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

```
int main(void) {  
    int a = 2000000000;  
    int b = 2000000000;  
    printf("%d + %d = %d\n", a, b, a+b);  
}
```



```
gcc -o test2.c test2.c
```

2000000000 + 2000000000 = -294967296

↑
Σ
↑
10



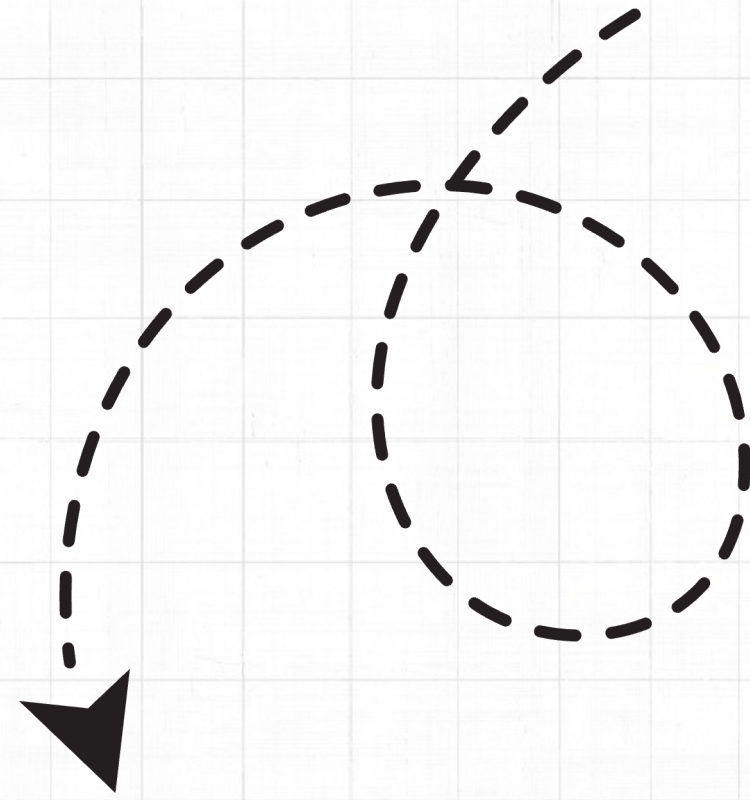
```
#include <stdio.h>
```

```
int main(void) {  
    size_t a = 2000000000;  
    size_t b = 2000000000;  
    printf("%zu + %zu = %zu\n", a, b, a+b);  
}
```



```
gcc -o tset2.c test2.c
```

2000000000 + 2000000000 = 4000000000



Questions?



Thank You

- <https://lacti.github.io/2011/01/08/different-between-size-t-ssize-t/>
- <https://code4human.tistory.com/119>
- <https://blog.naver.com/luexr/223281246126>
- <https://olivertree-cs.tistory.com/entry/sizet-%EC%9E%90%EB%A3%8C%ED%98%95-%EC%A3%BC%EC%9D%98%EC%A0%90>