ft_printf by sujo

[format]

- zero : 0 플래그, **init = 0**
- left : - 플래그, **init = 0**
- width : 폭, **init = 0**
- dot : '.', **init = 0**
- precision : 정밀도, **init = -1**

※ precision의 초깃값이 –1인 이유는 precision이 아예 안 올 때와 값이 0일 때의 결과가 다르기 때문.

※ 우선순위 : - > 0

1) 형식 지정자 기본 구조

%[-, 0][*, num].[*, num]TYPE

2) 형식 지정자

- **%c** : 문자형 char
- **%s** : 문자열 char *
- **%d** : 10진수 정수(int)
- **%i** : %d와 동일
- **%x** : 16진수 정수 lower (unsigned int)
- **%X** : 16진수 정수 upper (unsigned int)
- **%p** : 주소값 (void *)

3) flag

- **-** : 왼쪽 정렬
- **0** : 형식지정자가 숫자이면서 오른쪽 정렬 일 때, 빈 공간에 '0' 삽입
- **.** : width와 precision의 구분
- ***** : 숫자 삽입이 가능한 와일드카드(?)

4) printf's return

출력되는 총 길이를 return.

5) format setting

['-'가 들어온 경우]

- left를 1로 세트
- zero는 0으로 세트

['0'가 들어온 경우]

- zero를 1으로 세트
- left가 1인 경우 zero는 0으로 세트

['*'가 들어온 경우]

- **width 위치일 때,**
  - ✓ 인자로 받아온 값이 양수라면 width 값으로 세트
  - ✓ 인자로 받아온 값이 음수라면 left를 1로 세트하고, -1을 곱하여 width 값으로 세트
  - ✓ 인자로 받아온 값이 0 이라면 zero를 1로 세트
- **precision 위치일 때,**
  - ✓ 인자로 받아온 값이 양수라면 precision 값으로 세트
  - ✓ 인자로 받아온 값이 음수라면 precision을 –2로 세트 (후에 p일 경우 처리를 해주기 위함)

['.'가 들어온 경우]

- dot을 1로 세트
- precision을 0으로 세트

※ 이것을 기준으로 width와 precision을 구분

[num, 숫자]

- dot을 기준으로 각각 width 또는 precision에 세트
- 단, precision에 음수는 허용되지 않음.

ft_printf by sujo

[%c]

- 문자를 출력하는 형식 지정자.
- char

1) 특징

- precision 무시
- 0 flag : undefined behaviors

2) algorithm

if. width > 1
    length = width
else.
    length = 1

[left = 0 (오른쪽 정렬)]

1. length – 1 만큼 ' ' 출력
2. 문자 출력

[left = 1 (왼쪽 정렬)]

1. 문자 출력
2. length – 1 만큼 ' ' 출력

3) return

if. width > 1
    return width
else.
    return 1

[%s]

- 문자열을 출력하는 형식 지정자
- char *

1) 특징

- precision이 존재한다면, 문자열이 그보다 길다고 해도 precision 만큼만 출력
- 받아온 문자열이 NULL 이라면 출력할 문자열은 "(null)"로 세트됨
- 0 flag : undefined behaviors

2) algorithm

***precision 재설정**
if. precision > str_len || precision <= -1
    precision = str_len

[width <= precision]

1. 문자열 출력

[left = 0 (오른쪽 정렬)]

1. width – precision 만큼 ' ' 출력
2. 문자열 출력

[left = 1 (왼쪽 정렬)]

1. 문자열 출력
2. width – precision 만큼 ' ' 출력

3) return

if. width > precision
    return width
else.
    return precision

ft_printf by sujo

[%d, %i]

- 10진수 정수형 출력
- int

1) 특징

- precision < nbr_size 이라면 빈 공간은 '0'을 출력
  ex) printf("[%5.3d]", 1); => [001]
- num과 precision이 0인 경우, width만큼 공백만을 출력
- num이 음수일 때, '-'는 precision의 크기에 포함되지 않음

※ 음수일 때 '-' 는 nbr_size에 포함되지 않음

ex) -14 => nbr_size = 2

2) algorithm

***precision 재설정**

if. zero = 1 **and** precision <= -1
    precision = width
if. precision <= -1 **or** presicion < nbr_size
    precision = nbr_size
if. num < 0
    minus = 1
else.
    minus = 0

[num = 0 && precision == 0]

1. width 만큼 ' ' 출력

[left = 0 (오른쪽 정렬)]

1. width – precision – minus 만큼 ' ' 출력
2. num이 음수라면 '-' 출력
3. precision – nbr_size 만큼 '0' 출력
4. num 출력

[left = 1 (왼쪽 정렬)]

1. num이 음수라면 '-' 출력
2. precision – nbr_size 만큼 '0' 출력
3. num 출력
4. width – precision – minus 만큼 ' ' 출력

3) return

if. num = 0 **&&** precision == 0
    return width
if. width > precision
    return width
else
    return precision + minus

ft_printf by sujo

[%%]

- '%'를 출력

1) 특징

- 0 플래그 사용 가능
- precision 무시(?)

2) algorithm

if. width > 1
    length = width
else.
    length = 1

[left = 0 (오른쪽 정렬)]

1. '0' 또는 ' ' 출력
2. '%' 출력

[left = 1 (왼쪽 정렬)]

1. '%' 출력
2. length – 1 만큼 ' ' 출력

3) return

if. width > 1
    return width
else.
    return 1

[%u]

- 부호없는 10진수 정수형 출력
- unsigned int

1) 특징

- precision < nbr_size 이라면 빈 공간은 '0'을 출력
  ex) printf("[%5.3d]", 1); => [001]
- num과 precision이 0인 경우, width만큼 공백만을 출력

2) algorithm

***precision 재설정**
if. zero = 1 and precision <= -1
    precision = width
if. precision <= -1 or presicion < nbr_size
    precision = nbr_size

[num = 0 && precision == 0]

1. width 만큼 ' ' 출력

[left = 0 (오른쪽 정렬)]

1. width – precision만큼 ' ' 출력
2. precision – nbr_size 만큼 '0' 출력
3. num 출력

[left = 1 (왼쪽 정렬)]

1. precision – nbr_size 만큼 '0' 출력
2. num 출력
3. width – precision – minus 만큼 ' ' 출력

3) return

if. num = 0 **&&** precision == 0
    return width
if. width > precision
    return width
else
    return precision

ft_printf by sujo

[%x, %X]

- 부호없는 16진수 정수형 출력
- unsigned int

1) 특징

- precision < nbr_size 이라면 빈 공간은 '0'을 출력
- num과 precision이 0인 경우, width만큼 공백만을 출력

2) algorithm

***precision 재설정**

if. zero = 1 **and** precision <= -1
    precision = width
if. precision <= -1 **or** presicion < nbr_size
    precision = nbr_size

[num = 0 && precision == 0]

1. width 만큼 ' ' 출력

[left = 0 (오른쪽 정렬)]

1. width – precision만큼 ' ' 출력
2. precision – nbr_size 만큼 '0' 출력
3. num 출력 (16진수)

[left = 1 (왼쪽 정렬)]

1. precision – nbr_size 만큼 '0' 출력
2. num 출력 (16진수)
3. width – precision – minus 만큼 ' ' 출력

3) return

if. num = 0 **&&** precision == 0
    return width
if. width > precision
    return width
else.
    return precision

[%p]

- 주소 출력
- void *
- 형 변환 시 unsigned long

1) 특징

- '0x'로 시작(nbr_size에 포함하기 +2)
- '0x'는 precision 크기에 포함되지 않음
- 주소값이 0(NULL)이면서 precisino이 0인 경우 주소부분을 출력하지 않음. ex) "0x"

2) algorithm

***precision, width 재설정**

if. precision < -1 **or** dot != 1 **or** precision < nbr_size
    precision = nbr_size
if. width < precision + 2
    width = precision + 2

※여기서 +2 는 '0x'의 글자 수

[left = 0 (오른쪽 정렬)]

1. width – precision – 2 만큼 ' ' 출력
2. "0x" 출력
3. precision – nbr_size 만큼 '0' 출력
4. 16진수 주소값 출력

[left = 1 (왼쪽 정렬)]

1. "0x" 출력
2. precision – nbr_size 만큼 '0' 출력
3. 16진수 주소값 출력
4. width – precision – 2 만큼 ' ' 출력

3) return

return width