PRINTF

Because putnbr and putstr aren't enough

putnbr와 putstr으로는 만족할 수 없기 때문에

Summary: This project is pretty straight forward. You will recode printf. Hopefully you will be able to reuse it in future projects without the fear of being flagged as a cheater.

요약 : 이 프로젝트는 꽤 단순합니다. 여러분은 printf 함수를 직접 구현하시면 됩니다. 희망컨대 여러분들은 cheater로 지목될 수 있다는 두려움 없이 추후 프로젝트에서 이것을 재활용할 수 있습니다.

You will mainly learn how to use variadic arguments.

여러분은 주로 가변 인자 (variadic arguments) 를 사용하는 방법에 대해 배울 것입니다.

CONTENTS

CHAPTER 1

INTRODUCTION

The versatility of the printf function in C represents a great exercise in programming for us. This project is of moderate difficulty. It will enable you to discover variadic functions in C.

`C` 에서 `printf` 함수의 다재다능함은 프로그래밍에 있어 우리에게 훌륭한 연습이 됩니다. 이 프로젝트는 중간 정도의 난이도를 가지며, 여러분들이 `C` 에서 가변 함수들을 배울 수 있도록 도와줍니다.

The key to a successful ft_printf is a well-structured and good extensible code.

성공적인 `ft_printf` 의 핵심은 체계적이고 확장성 있는 코드입니다.

CHAPTER 2

COMMON INSTRUCTIONS

- *Your project must be written in accordance with the Norm. If you have bonus files/functions, they are included in the norm check and you will receive a 0 if there is a norm error inside.*

  프로젝트는 Norm 규칙에 맞춰 작성되어야 합니다. 보너스 파일/함수가 존재할 경우, 그 또한 norm 검사에 포함되며 norm error가 있을 시 0점을 받게 됩니다.

- *Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a 0 during the evaluation.*

  함수들은 정의되지 않은 행동들과 별개로 예기치 않게 중단되어서는 안 됩니다.(예를 들어, segmentation fault, bus error, double free 등) 만약 이렇게 중단되면, 당신의 프로젝트는 작동하지 않는 것으로 여겨지고 평가에서 0점을 받을 것입니다.

- *All heap allocated memory space must be properly freed when necessary. No leaks will be tolerated.*

  필요한 경우 heap에 할당된 모든 메모리 공간은 적절하게 해제되어야 합니다. 메모리 누수는 용납될 수 없습니다.

- *If the subject requires it, you must submit a Makefile which will compile your source files to the required output with the flags -Wall, -Wextra and -Werror, and your Makefile must not relink.*

  과제에서 필요한 경우, **-Wall -Wextra -Werror** 플래그를 지정하여 컴파일을 수행하는 **Makefile**을 제출해야 합니다. Makefile은 relink 되어서는 안 됩니다.

- *Your Makefile must at least contain the rules $(NAME), all, clean, fclean and re.*

  Makefile은 최소한 **$(NAME), all, clean, fclean, re** 규칙을 포함해야 합니다.

- *To turn in bonuses to your project, you must include a rule bonus to your Makefile, which will add all the various headers, librairies or functions that are forbidden on the main part of the project. Bonuses must be in a different file _bonus.{c/h}. Mandatory and bonus part evaluation is done separately.*

  프로젝트에 보너스를 제출하려면, Makefile에 **보너스 규칙**을 포함해야 합니다. 이 보너스 규칙은 프로젝트의 메인 파트에서 금지되었던 모든 다양한 헤더, 라이브러리, 또는 함수들을 추가하여야 합니다. 보너스 과제는 반드시 _**bonus**.{c/h}라는 별도의 파일에 있어야 합니다. 반드시 수행하여야 하는 메인 파트의 평가와 보너스 파트의 평가는 별도로 이뤄집니다.

- *If your project allows you to use your libft, you must copy its sources and its associated Makefile in a libft folder with its associated Makefile. Your project's Makefile must compile the library by using its Makefile, then compile the project.*

  만일 프로젝트에서 여러분의 libft 사용을 허용한다면, 소스들과 관련 Makefile을 함께 루트 폴더 안에 있는 libft 폴더에 복사해야 합니다. 프로젝트의 Makefile은 우선 libft의 Makefile을 사용하여 라이브러리를 컴파일한 다음, 프로젝트를 컴파일해야 합니다.

- *We encourage you to create test programs for your project even though this work won't have to be submitted and won't be graded. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.*

  이 과제물을 제출할 필요가 없고, 채점 받을 필요가 없을지라도, 저희는 여러분들이 프로젝트를 위한 테스트 프로그램을 만들 것을 권장합니다. 이것은 여러분의 과제물과 동료들의 과제물을 쉽게 테스트할 수 있게 도울 것입니다. 또한, 평가를 진행할 때 이러한 테스트 프로그램들이 특히 유용하다는 사실을 알게 될 것입니다. 평가 시에는 여러분의 테스트 프로그램과 평가 받는 동료의 테스트 프로그램들을 당연히 자유롭게 사용할 수 있습니다.

- *Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.*

  할당된 git 저장소에 과제물을 제출하세요. 오직 git 저장소에 있는 과제물만 등급이 매겨질 것입니다. Deepthought가 평가하는 과제의 경우엔, 동료평가 이후에 Deepthought가 수행됩니다. 만약 Deepthought 평가 중에 오류가 발생한다면, 그 즉시 평가는 중지될 것입니다.

CHAPTER 3

MANDATORY PART

프로그램 이름	**libftprintf.a**
제출할 파일	*.c, */*.c, *.h, */*.h, Makefile
Makefile 규칙	all, clean, fclean, re, bonus
사용가능한 외부 함수	malloc, free, write, va_start, va_arg, va_copy, va_end
직접 만든 libft	사용 가능
설명	실제 printf의 동작을 모방한 ft_printf를 포함하는 라이브러리를 작성하세요.

- *The prototype of ft_printf should be* `int ft_printf(const char *, ...);`

  ft_printf의 프로토타입은 `int ft_printf(const char *, ...);` 이어야 합니다.

- *You have to recode the* `libc` *'s* `printf` *function*

  여러분은 `libc` 의 `printf` 함수를 재구현해야 합니다.

- *It must not do the buffer management like the real printf*

  실제 `printf` 처럼 버퍼 관리를 수행해서는 안 됩니다.

- *It will manage the following conversions: cspdiuxX%*

  다음 서식 지정자를 구현하세요 : `cspdiuxX%`

- *It will be compared with the real printf*

  실제 printf와 비교하여 채점할 것입니다.

- *You must use the command* `ar` *to create your librairy, using the command* `libtool` *is forbidden.*

  `ar` 명령어를 이용하여 라이브러리를 만들어야 합니다. `libtool` 을 사용하는 것은 금지됩니다.

A small description of the required conversion:

필요한 서식 지정자에 대한 간단한 설명입니다:

- *%c print a single character.*

  %c는 단일 문자 (character) 한 개를 출력합니다.

- *%s print a string of characters.*

  %s는 문자열 (string) 을 출력합니다.

- *%p The void * pointer argument is printed in hexadecimal.*

  %p는 void * 형식의 포인터 인자를 16진수로 출력합니다.

- *%d print a decimal (base 10) number.*

  %d는 10진수 숫자를 출력합니다.

- *%i print an integer in base 10.*

  %i는 10진수 정수를 출력합니다.

- *%u print an unsigned decimal (base 10) number.*

  %u는 부호 없는 10진수 숫자를 출력합니다.

- *%x print a number in hexadecimal (base 16).*

  %x는 숫자를 16진수로 출력합니다.

- *%% print a percent sign.*

  %%는 퍼센트 기호 (%) 를 출력합니다.

> 💡
>
> *for more complete references :* `man 3 printf / man 3 stdarg`
>
> 더 완벽한 참고 자료는 `man 3 printf / man 3 stdarg`

CHAPTER 4

BONUS PART

- *If the Mandatory part is not perfect don't even think about bonuses*

  필수 구현 파트가 완벽하지 않으면, 보너스는 생각도 하지 마세요.

- *You don't need to do all the bonuses*

  모든 보너스를 구현할 필요는 없습니다.

- *Manage any combination of the following flags:* `'-0.'` *and minimum field width with all conversions*

  다음 플래그들의 조합 (any combination) 을 구현하세요 : `'-0.'`, 그리고 각 서식 지정자별 최소 폭

- *Manage all the following flags:* `'# +'` *(yes, one of them is a space)*

  다음 플래그를 모두 구현하세요 : `'# +'` (맞아요, 한 개는 공백이에요)

> 💡
>
> *If you plan to do bonuses you should think about how to do them from the beginning to avoid a naive approach.*
>
> 보너스를 구현하실 예정이라면, 단순하게 접근하지 않기 위하여 처음부터 어떻게 구현을 해야 할 지 고민해 보셔야 할 겁니다.