Introduction to Operating Systems

Homework 1: Simple Preprocessor

Due 9/28/2024 @ 11:59pm

Introduction

In this homework we will build a simple preprocessor… somewhat similar to the one that is inside in the C language, but simpler!

The C preprocessor is controlled by “directives”. Directives are lines which start with `#`.

There are many directives… if you are curious you can read about them at here.
In this homework we will primarily focus on simple `#define` without parameters.

Here is an example C program:

```C/C++
#include <stdio.h>

#define x 42

int main() {
    printf("The meaning of life: %d\n", x);
}
```

If you compile and and run this:

```Unset
$ gcc test.c -o test
$ ./test
The meaning of life: 42
```

If we remove the definition of `x`, the program will stop building:

```C/C++
#include <stdio.h>

int main() {
    printf("The meaning of life: %d\n", x);
}
```

```Unset
$ gcc test.c -o test
test.c:4:44: error: use of undeclared identifier 'x'
    printf("The meaning of life is: %d\n", x);
                                           ^
1 error generated.
```

But we can provide a new definition of `x` on the command line and it will work once again:

```Unset
$ gcc test.c -o test -Dx=43
$ ./test
The meaning of life is: 43
```

In this assignment, to keep things simple, we will only do the command line version of this…
The goal would be to write a program, `preprocess` which you can compile and execute as follows:

```Unset
$ proprocess test.c -Dx=43
int foo {
```

```
    return 43;
}
```

Here the highlighted text is the output.
This is roughly equivalent to `gcc -E -P test.c -Dx=43` which on my machine produces the same result.

```
Unset
$ gcc -E -P test1.c -Dx=43
int foo {
    return 43;
}
```

Rubric

The homework will be scored out of 100 points, but with extra credits people can score as much as 150!

Task 1 (20pt): Hello, World!

In this task you will have to implement, compile, and execute hello world inside xv6!

We need the following result from inside xv6:

```
Unset
$ ./hello
Hello, World!
```

Task 2 (60pt): Simple Preprocessor

You need to write a program called `preprocess` which accepts a text file as its first parameter and then a variable number of definitions on the command line. The program needs to then read the text file, substitute all variables in the code with their definitions and print the result. The program should be executable as:

`./preprocess <input_file> -D<var1>=<val1> -D<var2>=<val2> … -D<varN>=<valN>`

Variables will be valid C identifiers. Values can be anything!

The output of your program should be consistent with the output of `gcc -E -P <input_file> -D…`

Here are some examples. Consider the following file saved as `data.txt`:

```
Unset
Hello, Kamen!
I am your AI assistant and I am here to help.
Please ask me all your questions!
```

Examples:

```
Unset
$ gcc -E -P data.txt -DKamen=Sam
Hello, Sam!
I am your AI assistant and I am here to help.
Please ask me all your questions!

$ gcc -E -P data.txt -DKamen=Sam -Dam=is
Hello, Sam!
I is your AI assistant and I is here to help.
Please ask me all your questions!

$ gcc -E -P - < data.txt -DKamen=Sam -Dam=is -DI=
```

```
Hello, Sam!
  is your AI assistant and is here to help.
Please ask me all your questions!

$ gcc -E -P - < data.txt "-Dask=do not ask" "-Dall=any of" "-Dam=am not"
Hello, Kamen!
I am not your AI assistant and I am not here to help.
Please do not ask me any of your questions!
```

Note that these are not all the tests we will run – you should make sure that your code works on any input that matches the problem description.

Task 3 (20pt): 10 extra tests

You should test your code on more examples, not just the ones we have listed above.

Provide 10 of your best examples that you tested with, but think that others will fail on. If you can come up with an example that your code works on but the majority of the class fails on, you will get 5pt extra credit. This is in addition to the 20pt for this task.

Extra Credit Tasks (50pt):

Task 4 (10pt): Implement `#define <var_name> <value>`

Task 5 (10pt): Implement `#ifdef <var_name>`, `#else`, and `#endif`

Task 6 (20pt): Implement `#define <var_name>(<param>) <value>`

Task 7 (10pt): Implement `#include <filename>`

Extra Information

- TAs will communicate how to install and operate xv6 this week. You can play on a mac or linux computer you have access to before that if you want to start right away… or figure out how to do xv6 yourself :)
- You can work on this in pairs of two. I highly recommend it. If you don't have a partner, use google chat to coordinate and find one! Or message me.
- This is still a draft, which I wanted to get out early… If you have questions or suggestions, don't be shy. We will finalize the homework this week and it will officially be due on 9/28/2024!
    - TAs will communicate submission details.