Unix Assignment5

Team19 108072244 邱煒甯 108032053 陳凱揚 112065525 簡佩如

- Answer the question above. You need to explain why it is correct/incorrect in your report.

  The local variable `val` declared inside the `if` block is local to that block. Once the block completes its execution, `val` goes out of scope and the memory it was using on the stack can be used by other parts of the program. Since `ptr` is pointing to the address of this local variable `val`, dereferencing it outside the block (as in the `return` statement) leads to undefined behavior because you're accessing memory of a variable that is out of scope.

  To put it simply, the pointer `ptr` is pointing to a memory location that becomes invalid after the `if` block is exited. Accessing such memory can lead to unpredictable results, crashes, or other issues.

  To conclude, the code is incorrect due to the use of a pointer pointing to a local variable that goes out of scope before the pointer is dereferenced. If we want to return a pointer from a function, we should use the `malloc` function to allocate memory space, rather than returning the memory address of a local variable.
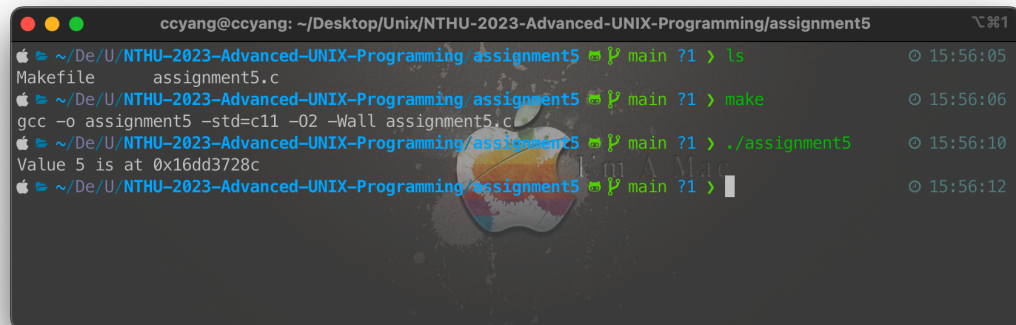
- Implement and modify the above C code to verify your answer.

```c
#include <stdio.h>

int* f1(int *val){
    int *address = val;
    return address;
}

int main(){
    int val = 5;
    int *address = f1(&val);
    printf("Value %d is at %p\n", *address, (void*)address);
    return 0;
}
```

Initially, we define a variable `val` and initialize it with the value 5 within the main function. Subsequently, we pass `&val` to the `f1` function. Inside `f1`, we get the

address of the `val` variable declared in the main function and then return this address. Upon receiving the address back in the main function, we proceed to print both the value and the address of the variable.

- Result