Memory Models & Debugging

Lecture 7
Jan 31st 2023 | COMP 211-002 | Joshua Bakita

Welcome!

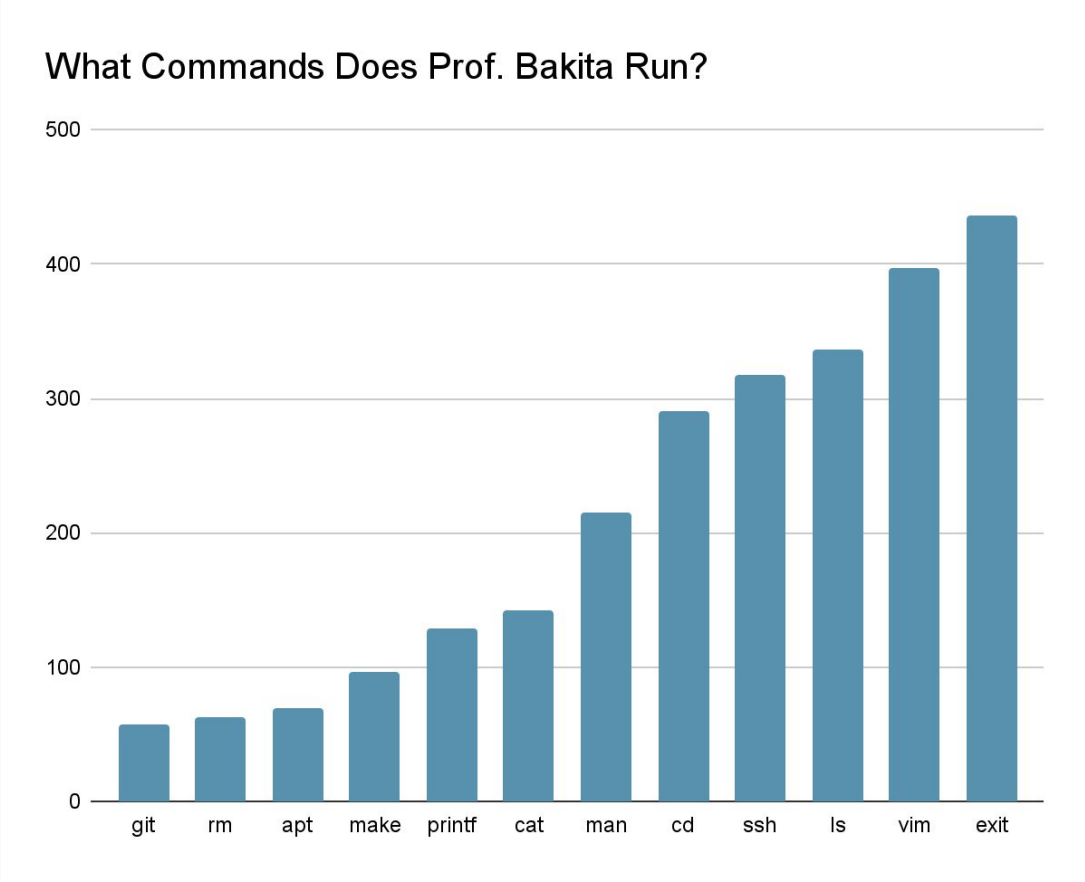
Today:

- ➔ Memory Models
- ➔ Debugging

Logistics:

- ➔ Assign 2 up.
- ➔ Avg. 87/90. Hand grading coming soon…

Interesting fact…



Logistics

Assignment 2 Overview

Built around tetris, as written by my friend Nathan Otterness (in three days).

Assignment by our excellent UTA, Ryan Good!

Two parts:

- ➔ Manually editing the tetris save file
- ➔ Building a program to edit the save file

The tetris program saves its data by dumping its internal TetrisGameState struct into a file.

The data in the file is an exact copy of the bytes stored in-memory that represent the struct.

This is how many programs, particularly those from the 1990s and early 2000s, write save files.

- ➔ Ex: Microsoft Word, Excel, etc.

Memory Models

One way of looking at last week's examples…

Memory Models

Three Sections of Memory

Static Memory
Global variables, `static` variables, string literals.

Stack Memory
Temporary variables for each function on the stack.

Heap Memory
Not used by default. Accessible via `malloc()`/`free()`-like functions

Question from Last Time: Unions

Thanks to UTA Kenan Poole for these slides!

Memory Models: Ex.

From last time…

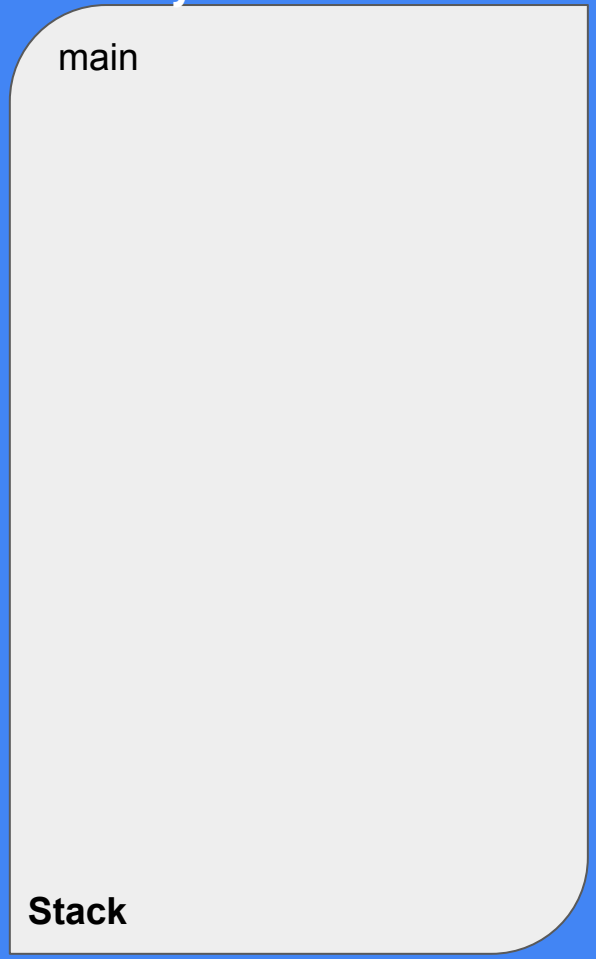
```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

What will this print for `i = 0`?

Try it yourself!

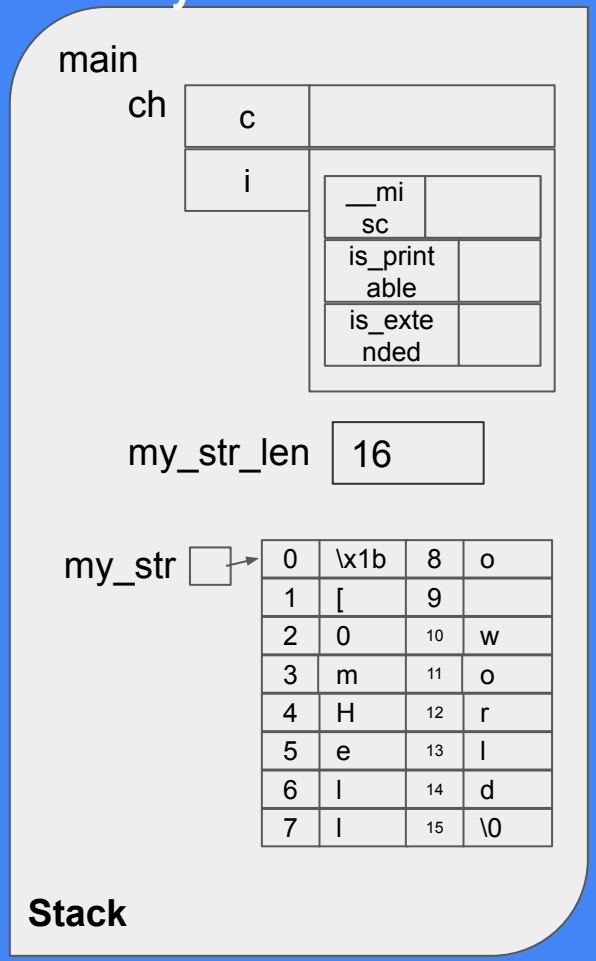
```
$ wget https://www.cs.unc.edu/~jbakita/teach/comp211-s23/l6/mystery2.c
$ gcc mystery2.c -o myst
$ ./myst
```

Memory Models: Ex.



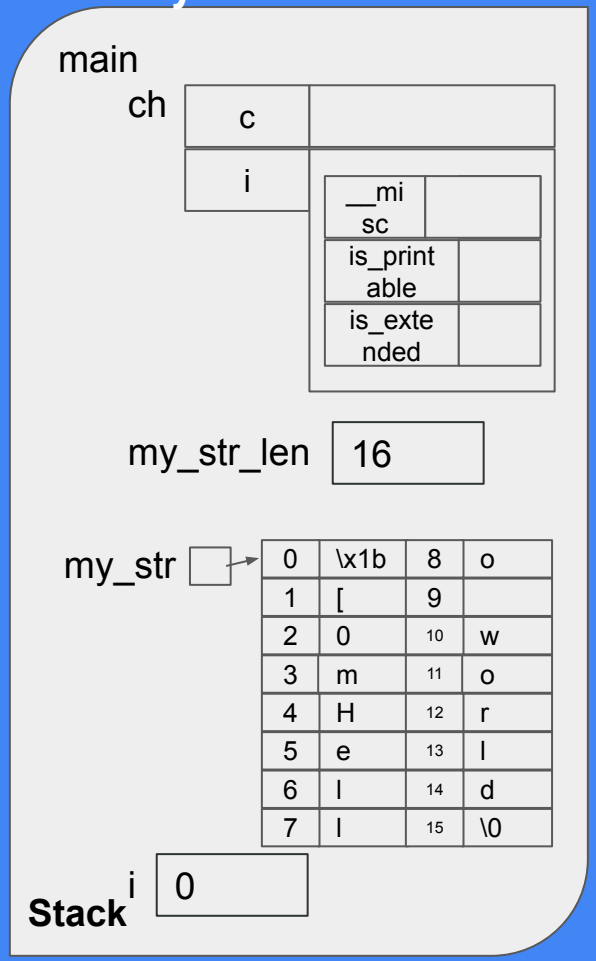

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
             printf("is printable.\n");
        } else {
             printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



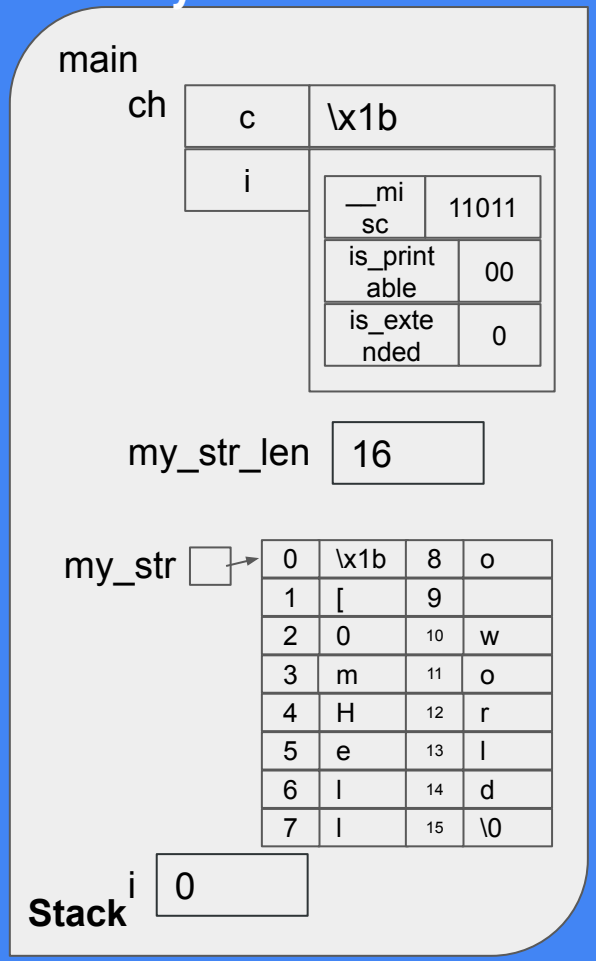

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.




```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

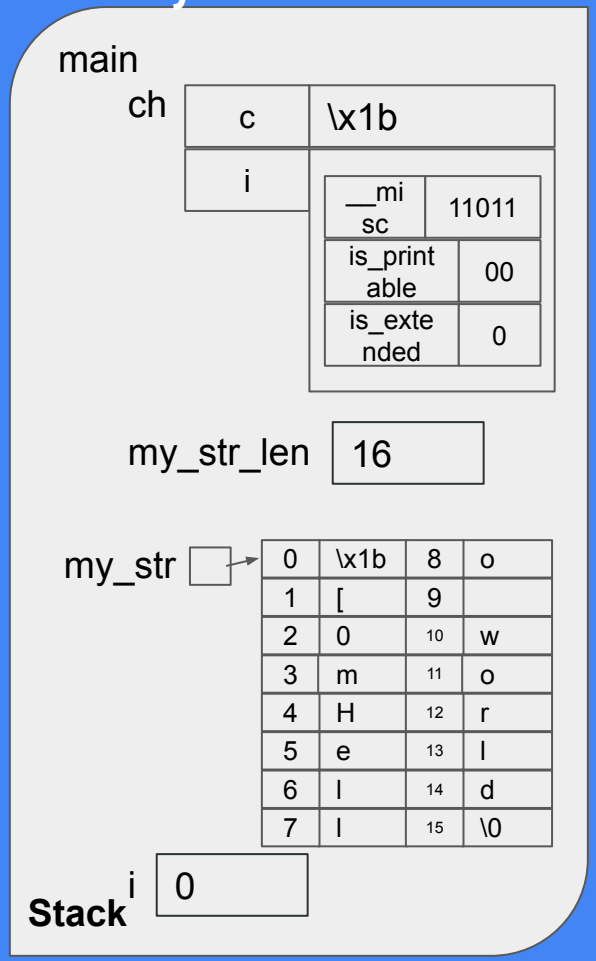
Memory Models: Ex.




```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

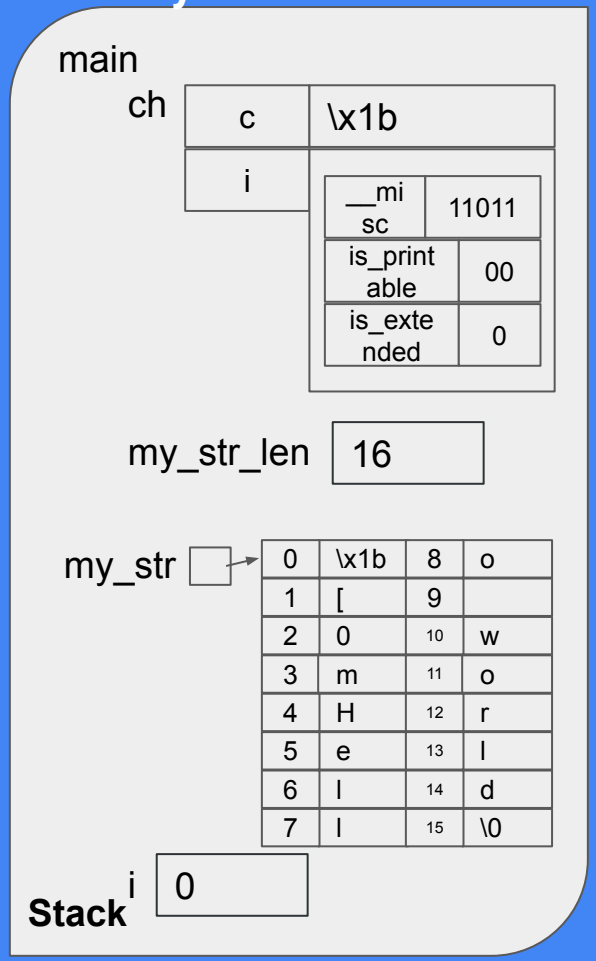
Note the type cast

Memory Models: Ex.



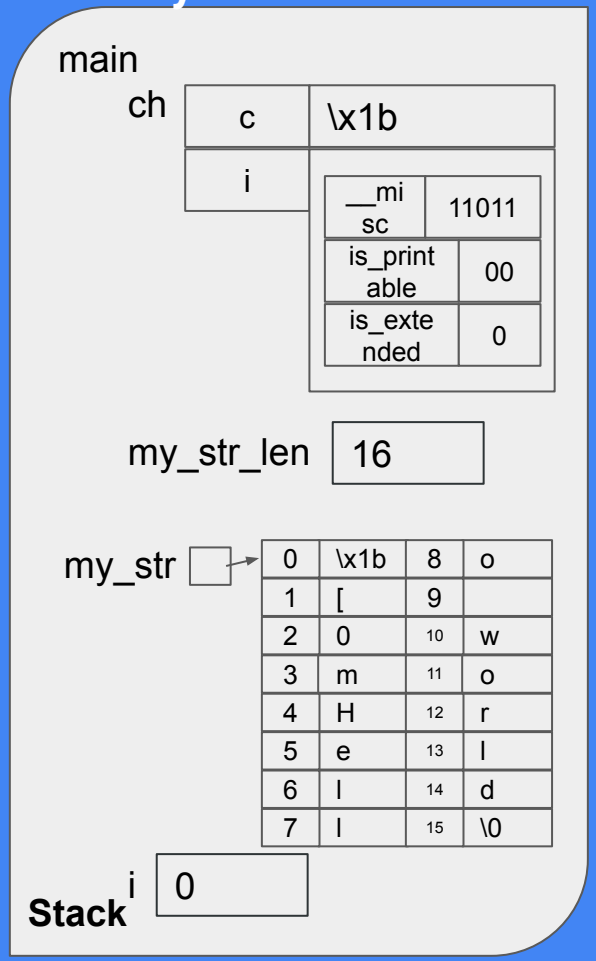

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



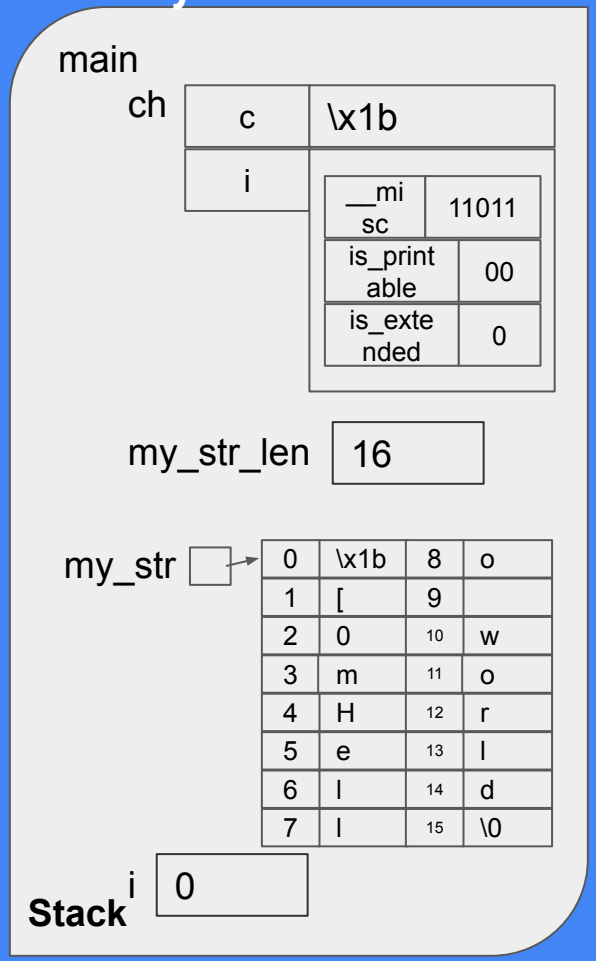

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



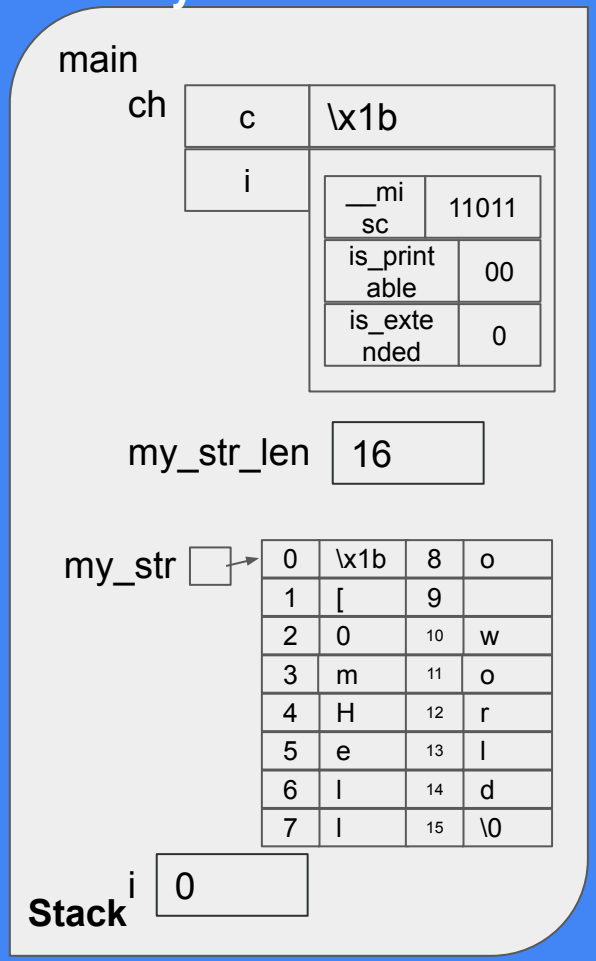

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



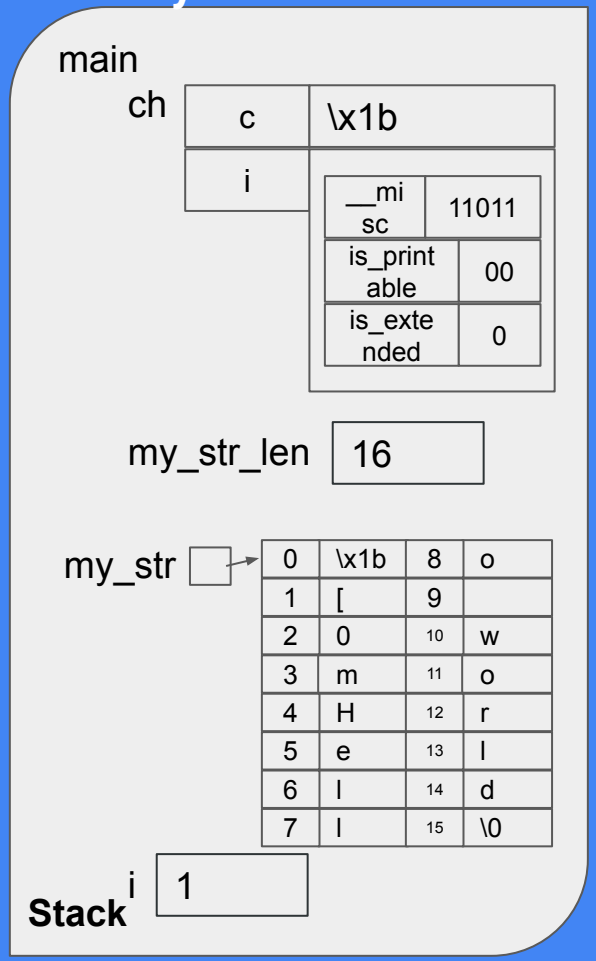

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



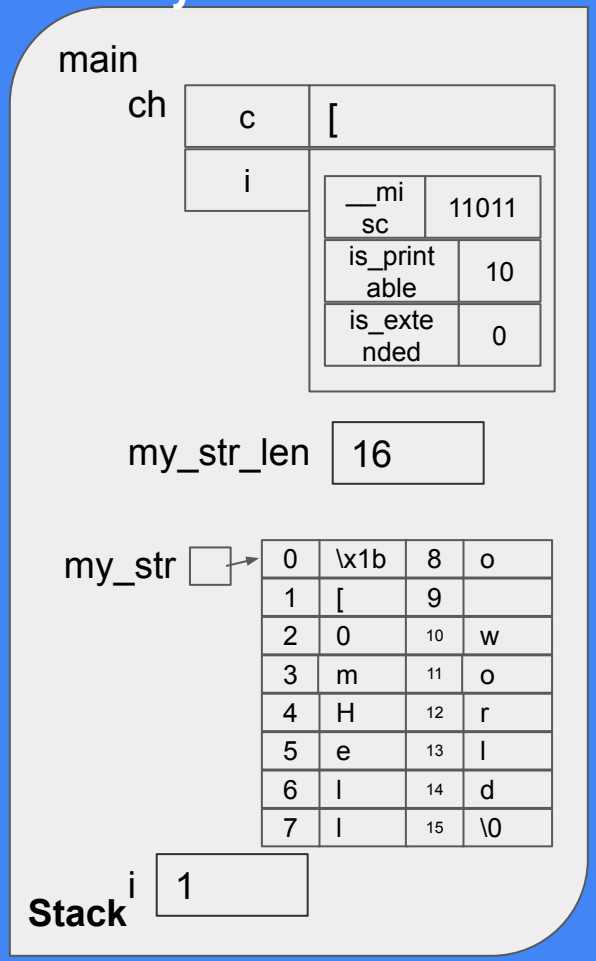

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



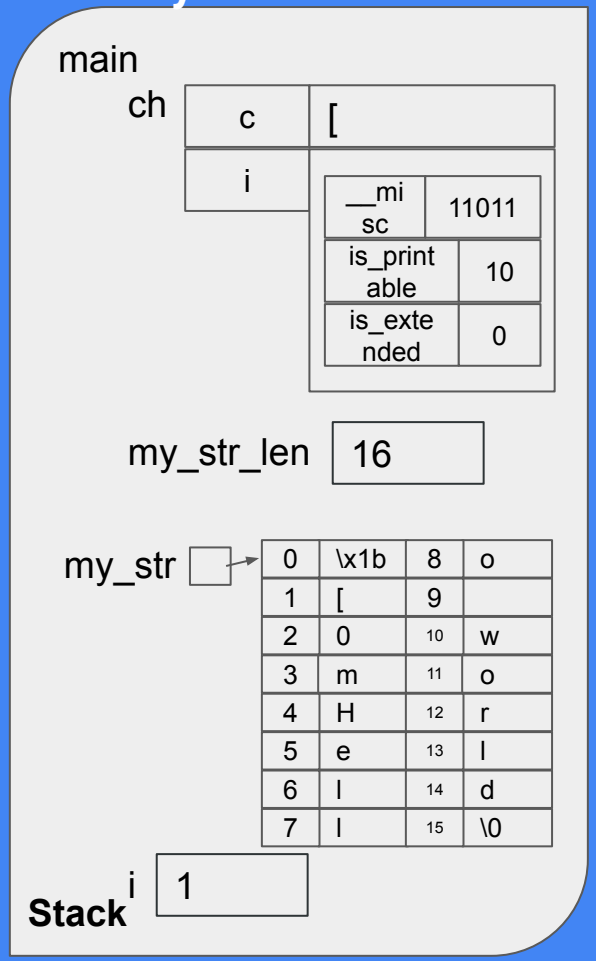

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



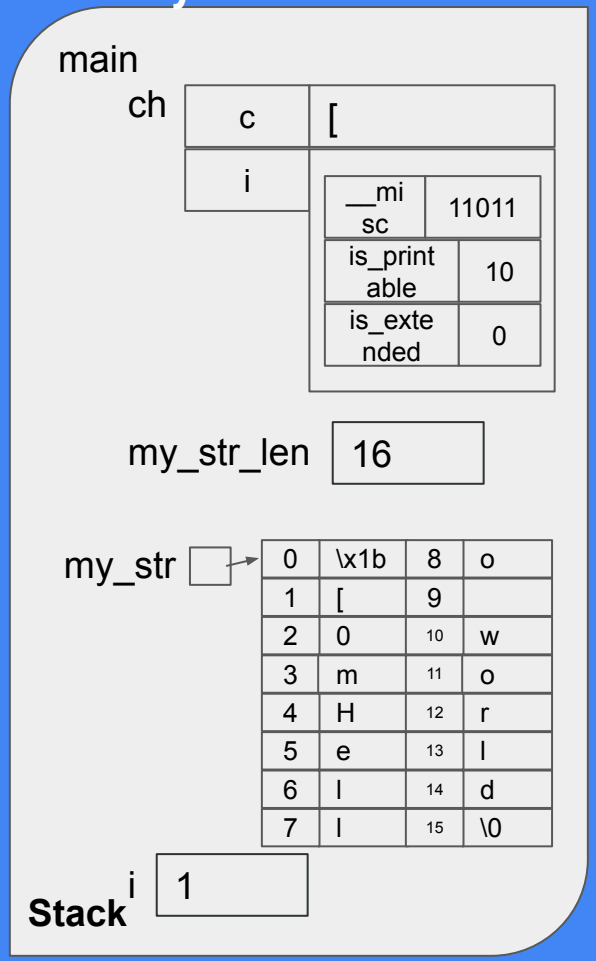

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



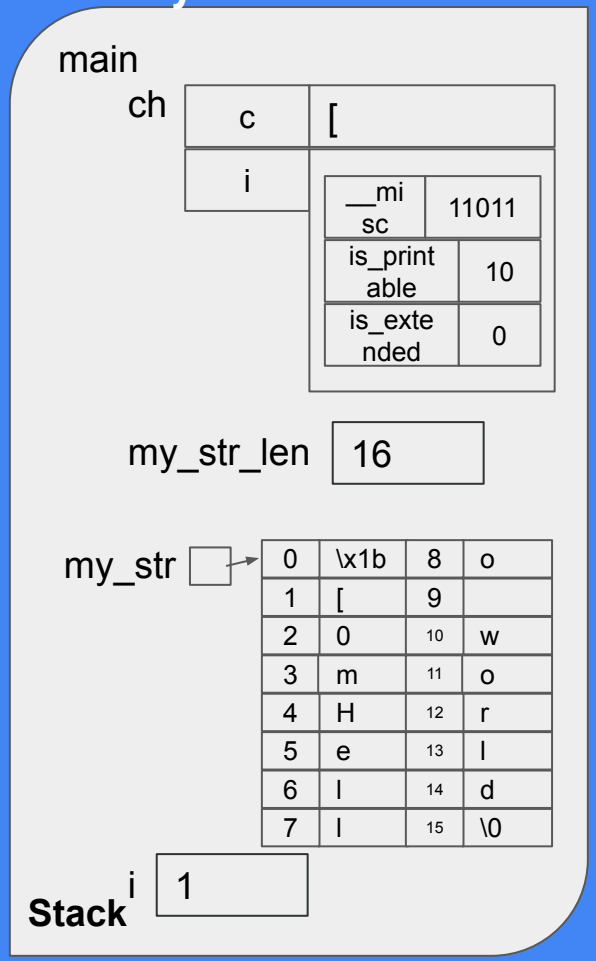

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



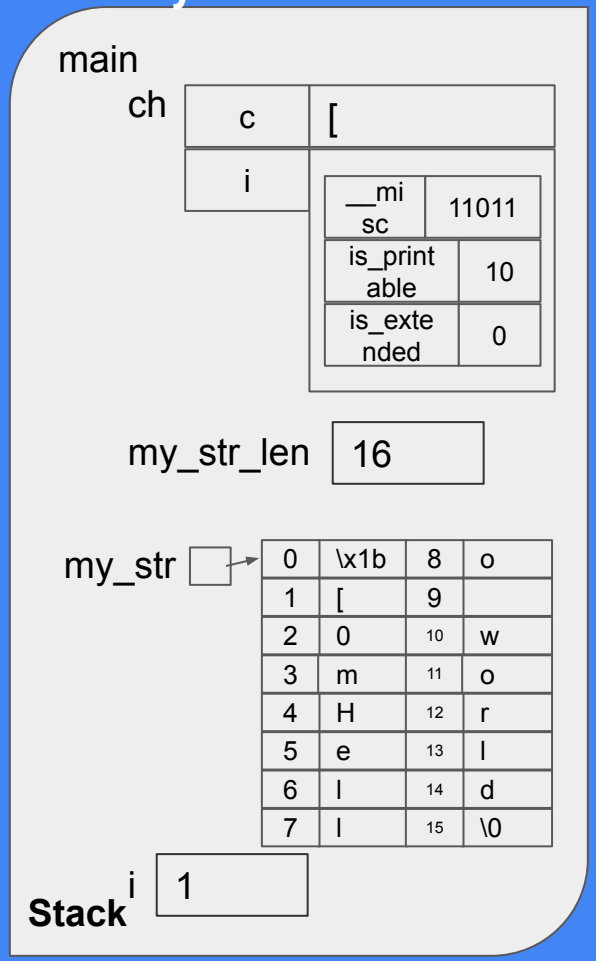

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



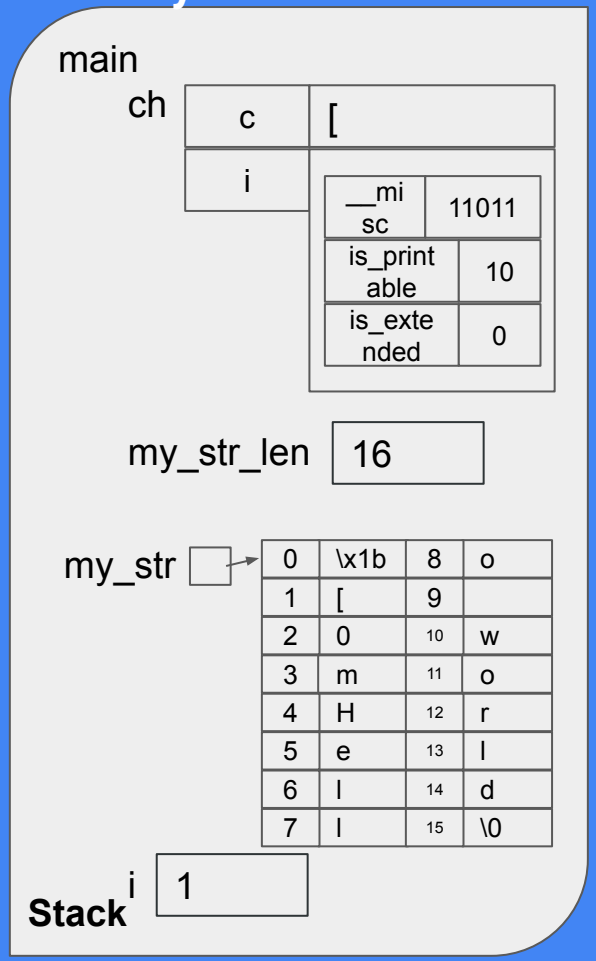

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



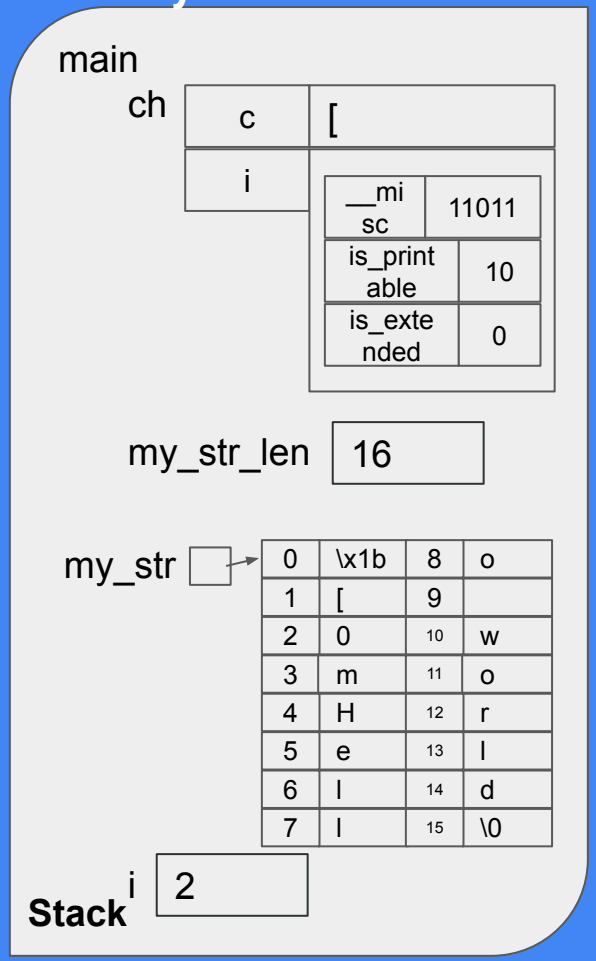

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.




```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Memory Models: Ex.



... for 14 more loops

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

Question from Last Week: Structs

Thanks to UTA Kenan Poole for these slides!

Memory Models Ex. 2

From last week…

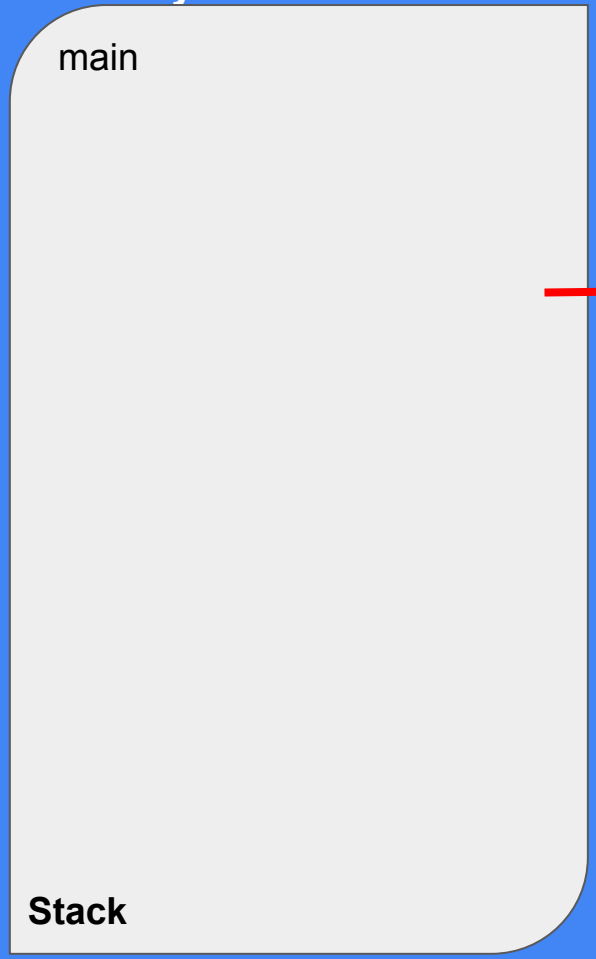
```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Which variable(s) will cause printf() to always print "UNC"?

Try it yourself!

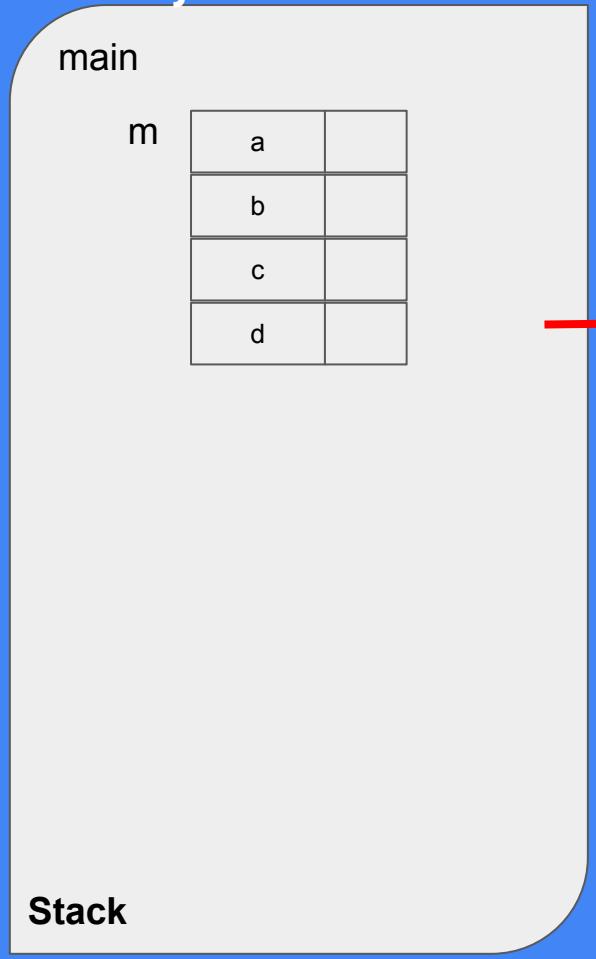
```
$ wget https://www.cs.unc.edu/~jbakita/teach/comp211-s23/l5/mystery.c
$ gcc mystery.c -o myst
$ ./myst
```

Memory Models Ex. 2



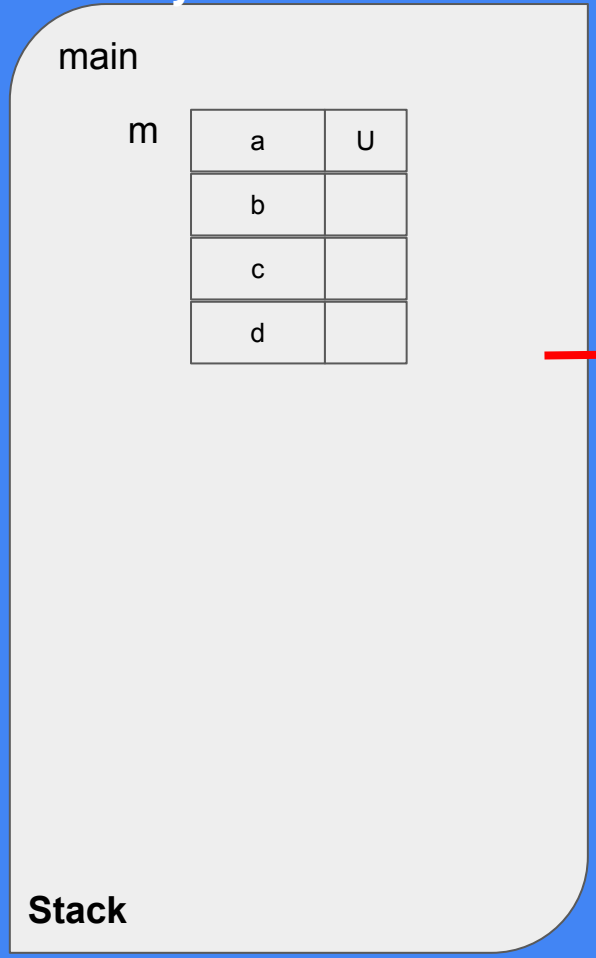

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2



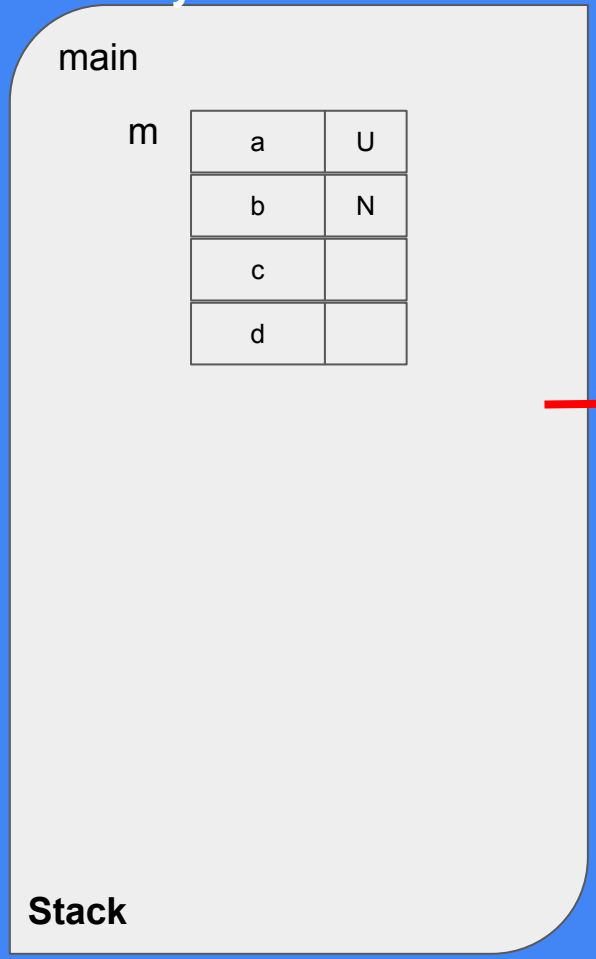

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2



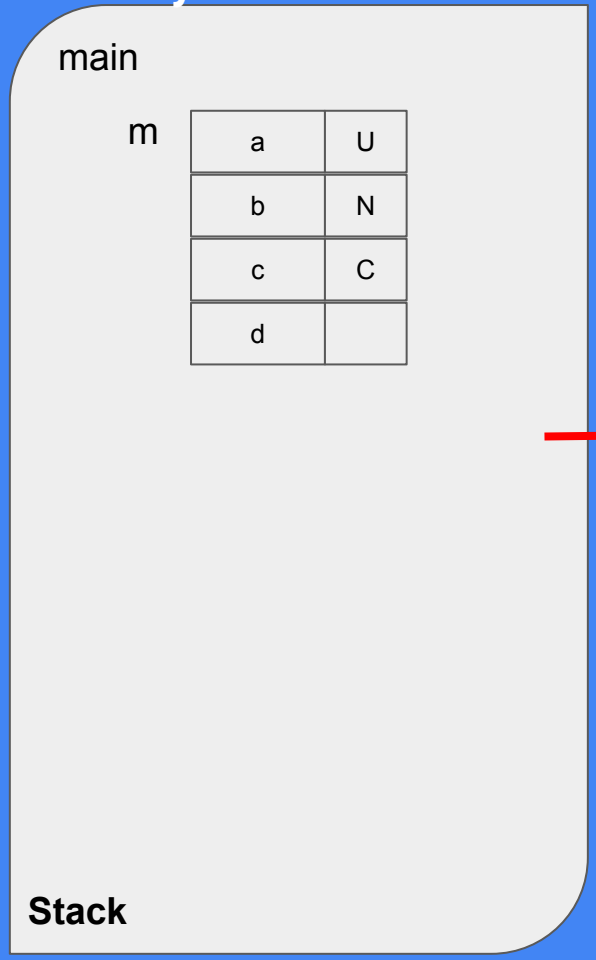

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2



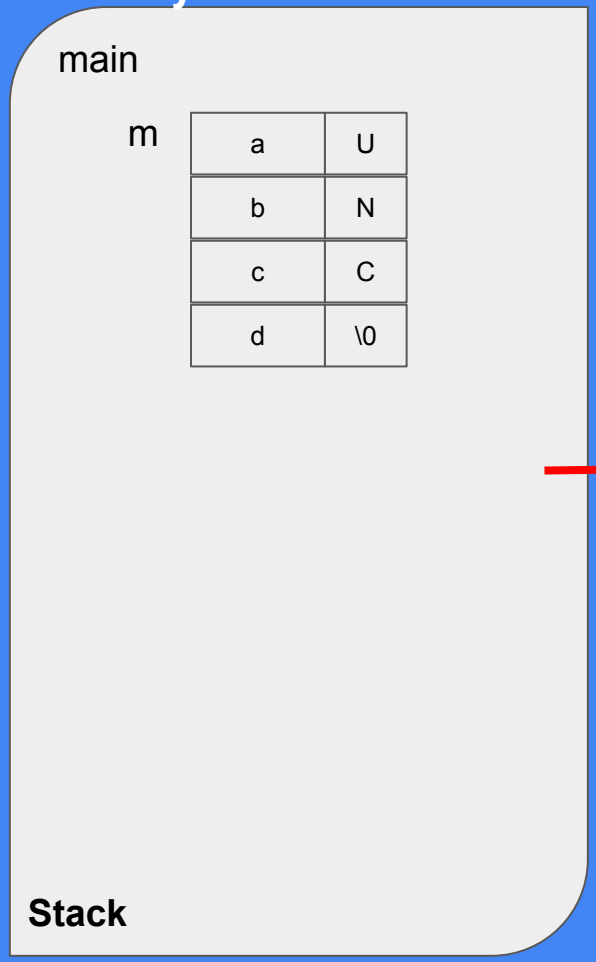

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2



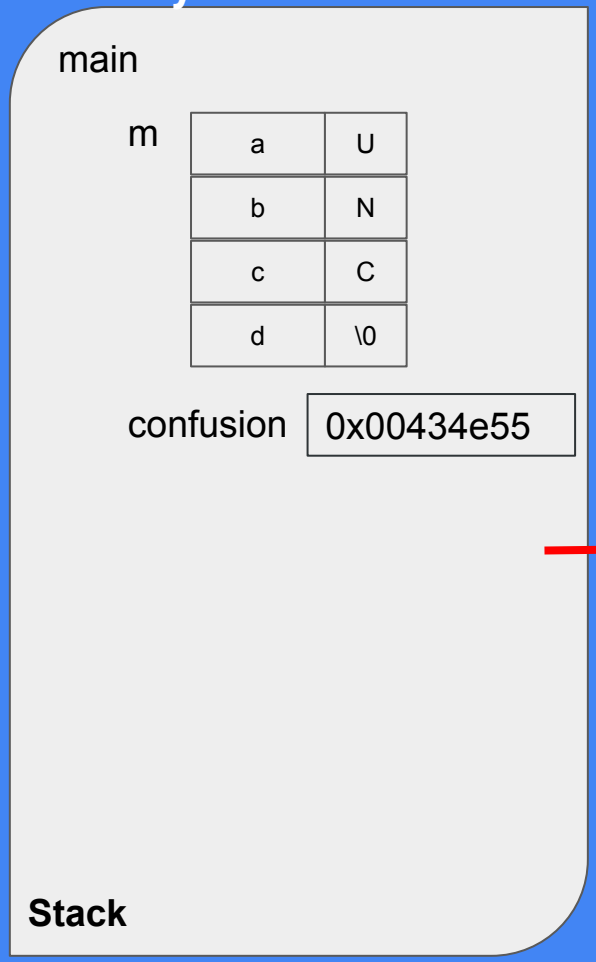

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2



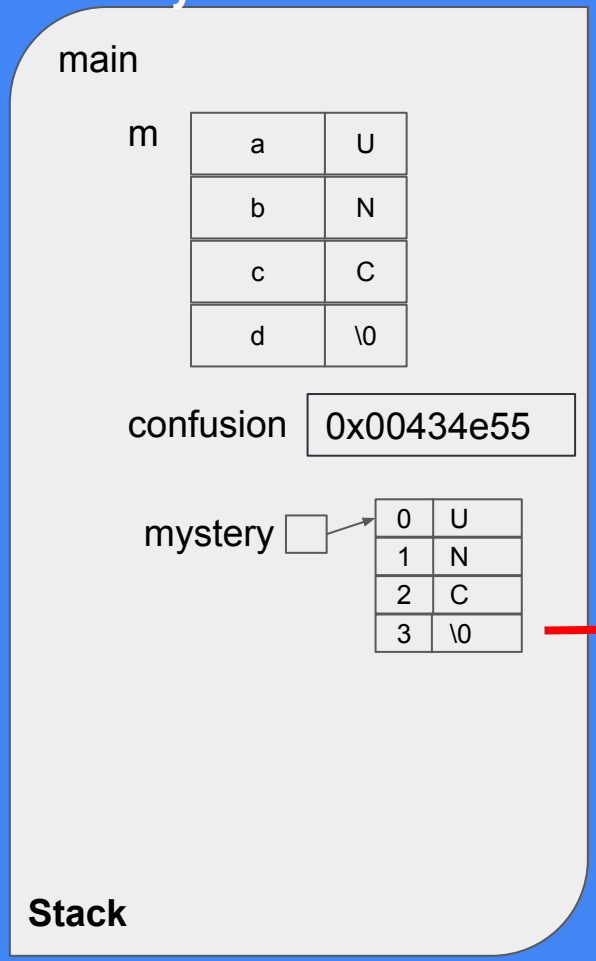

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2



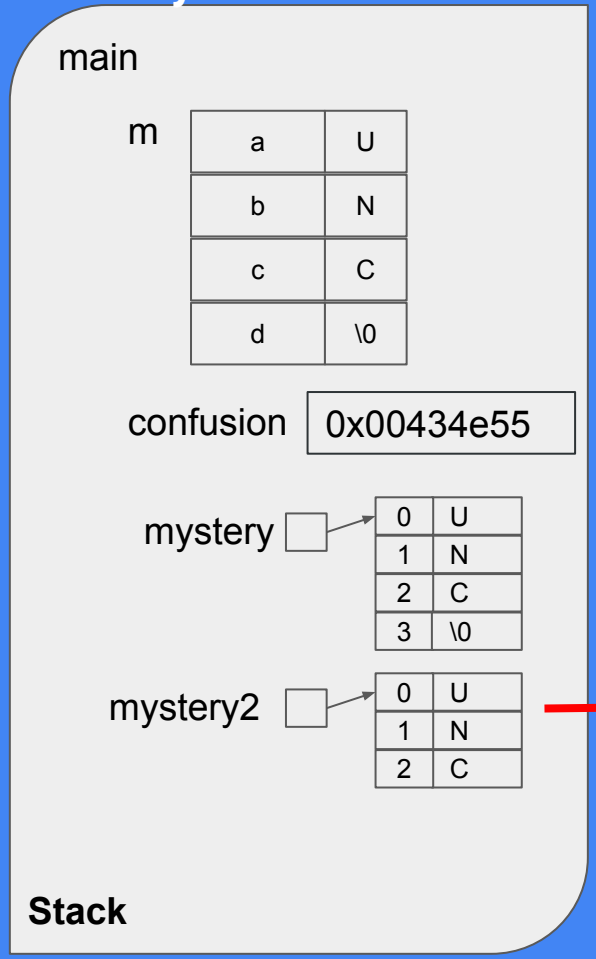

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2



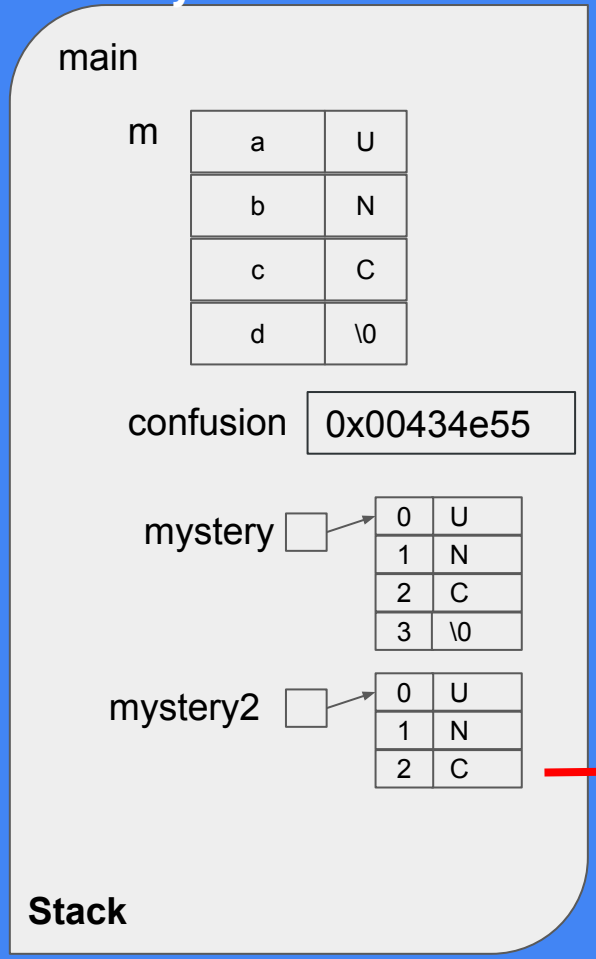

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2



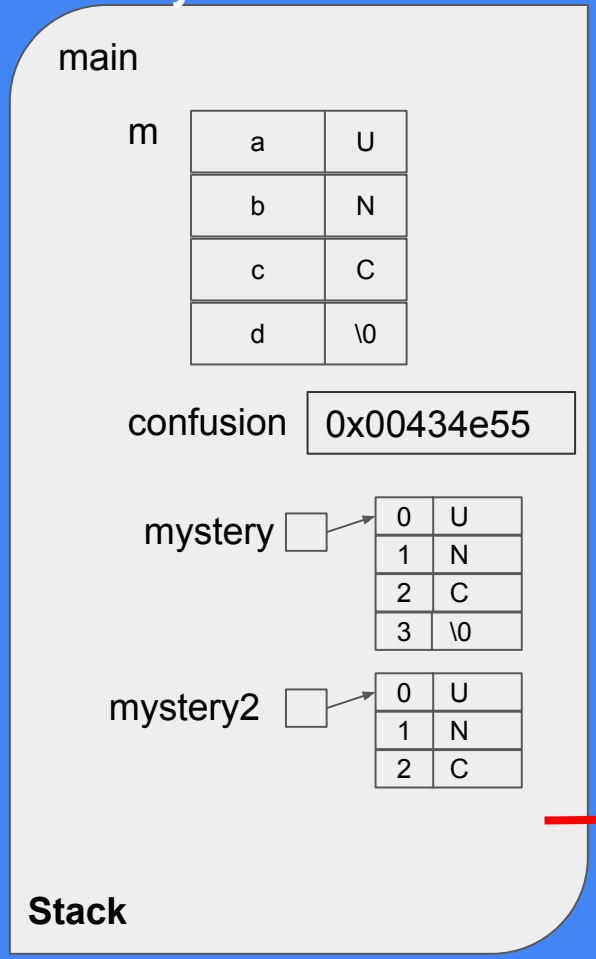

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2



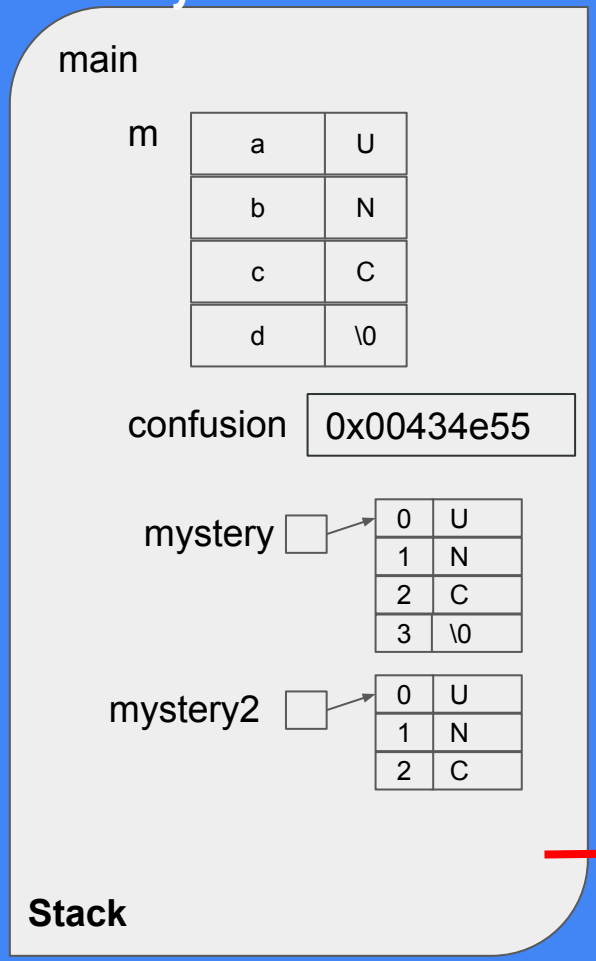

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2



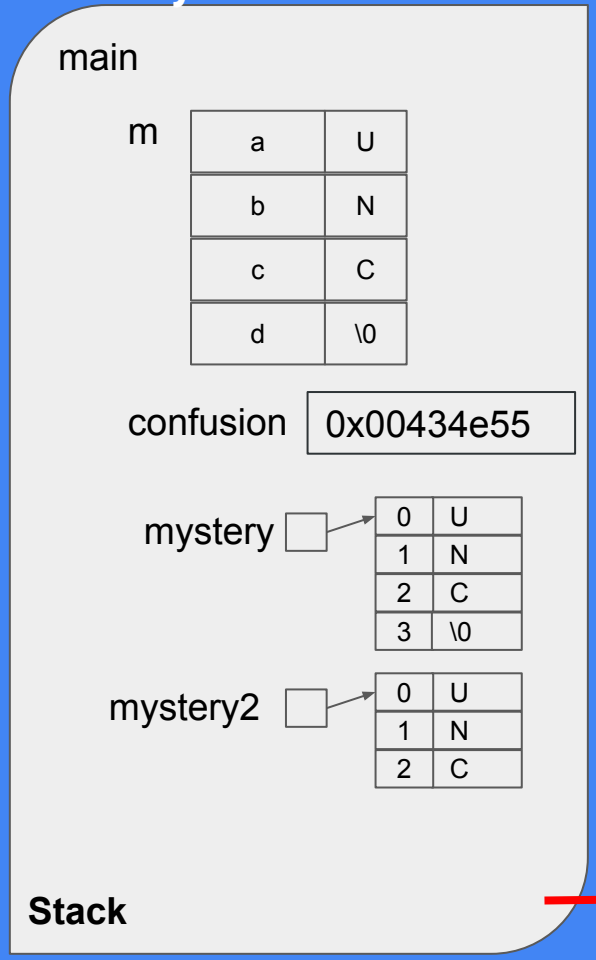

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2




```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2




```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Memory Models Ex. 2

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Debugging

Another way to look at last week's examples…

Command Line

`valgrind prog`	Run `prog` with valgrind
`gdb prog`	Start the GNU Debugger on `prog`
`info thing`	View detailed manual for `thing`
`xxd file`	Print `file` as hexadecimal
`wget addr`	Download file from `addr`
`rm file`	Delete `file`
`cd dir`	Move to `dir`
`cat file`	Print contents of `file`
`cp fileA fileB`	Copy `fileA` to `fileB`

Vim Commands (Normal Mode)

`dd`	Delete current line
`D`	Delete from cursor to end-of-line
`>>`	Increase indent
`<<`	Decrease indent
`O`	Add line above cursor and enter insert mode
`o`	Add line below cursor and enter insert mode

For Your ~/.vimrc Config File

```
set cindent
set nowrap
```

```
jbakita@jbakita:~/Source/comp524/COMP524FinalTests$ jdb people
Initializing jdb ...
> stop at people.main
Deferring breakpoint people.main.
It will be set after the class is loaded.
> run
run people
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
>
VM Started: Set deferred breakpoint people.main

Breakpoint hit: "thread=main", people.main(), line=3 bci=0
3    Person p0 = new Instructor();//"Jane");

main[1] step
>
Step completed: "thr                <init>(), line=21 bci
21    class Instructo

main[1] print 2+2
 2+2 = 4
main[1] step
>
Step completed: "thread=main", Person.<init>(), line=11 bci=0
11    class Person {

main[1] step
>
Step completed: "thread=main", Instructor.<in
21    class Instructor extends Person {

main[1]
```

Java has JDB

```
jbakita@comp211-2sp23:~$ python3 -m pdb /playpen/submit.py
> /playpen/submit.py(2)<module>()
-> import requests
(Pdb) break 81
Breakpoint 1 at /playpen/submit.py:81
(Pdb) run my_folder/gdtbath.c
Restarting /playpen/submit.py with arguments:
        my_folder/gdtbath.c
> /playpen/submit.py(2)<module>()
-> import requests
(Pdb) c
Welcome to the COMP 211-002, Spring 2023, Assignment 1 Upload
> /playpen/submit.py(81)<module>()
-> print('\033[3mThi                ave your credentials
')
(Pdb) bt
  /usr/lib/python3.8
-> exec(cmd, globals, locals)
  <string>(1)<module>()
> /playpen/submit.py(81)<module>()
-> print('\033[3mThis uploader does not save your credentials
')
(Pdb) p sys.argv
['/playpen/submit.py', 'my_folder/gdtbath.c']
(Pdb) step
This uploader does not save your credentials, and only uses t
                .py(82)<module>()
              > Gradescope over an encrypted (SSL) connect

                pe over an encrypted (SSL) connection.
> /playpen/submit.py(87)<module>()
```

Python has PDB

And they support a similar set of commands!

Memory Models Ex. 1

From last time…

```c
#include <stdio.h>

struct char_info {
    char __misc:5;
    char is_printable:2;
    char is_extended:1;
};

union e_char {
    char c;
    struct char_info i;
};

int main() {
    union e_char ch;
    int my_str_len = 16;
    char my_str[] = "\x1b[0mHello world";

    for (int i = 0; i < my_str_len; i++) {
        ch = (union e_char)my_str[i];

        printf("char '%#x' at index %d ", ch.c, i);
        if (ch.i.is_extended) {
            printf("does not have a well-agreed upon meaning, and ");
        } else {
            printf("has a well-agreed upon meaning, and ");
        }
        if (ch.i.is_printable) {
            printf("is printable.\n");
        } else {
            printf("is not printable.\n");
        }
    }
    return 0;
}
```

What will this print for `i = 0`?

Try it yourself!

```
$ wget
https://www.cs.unc.edu/~jbakita/teach/comp211-s23/l6/mystery2.c
$ gcc mystery2.c -o myst
$ ./myst
```

Memory Models Ex. 2

From last week…

```c
#include <stdio.h>

struct myst {
    char a, b, c, d;
};

int main() {
    struct myst m;
    m.a = 0x55;
    m.b = 0x4e;
    m.c = 0x43;
    m.d = 0x00;

    int confusion = 0x00434e55;

    char mystery[] = {0x55, 0x4e, 0x43, 0x00};

    char mystery2[] = {'U', 'N', 'C'};

    printf("The mystery string says: %s\n", mystery);
    printf("The 2nd mystery string says: %s\n", mystery2);
    printf("The mystery struct says: %s\n", &m);
    printf("The confusion int says: %s\n", &confusion);
}
```

Which variable(s) will cause printf() to always print "UNC"?

Try it yourself!

```
$ wget https://www.cs.unc.edu/~jbakita/teach/comp211-s23/l5/mystery.c
$ gcc mystery.c -o myst
$ ./myst
```

Key GNU Debugger (GDB) Commands

Full command name

Shorthand

Access the full GDB manual via `info gdb` on the command line

Control Flow

`backtrace`	`bt`	List all stack frames
`select <frame#>`	`sel`	Select a stack frame as your context
`next`	`n`	Execute the next line from your context
`step`	`s`	Execute one line
`list`	`l`	Print source code

Data

`print <expr>`	`p`	Execute expression and print result (can modify data)
`info locals`	`i lo`	Print value of every local variable in your stack frame
`x <addr>`	`x`	Print bytes at `addr` in memory
`whatis <expr>`	`wha`	Print type of `expr`

Breakpoints

`break <file>:<l>`
`break <function>`
`break <function> if <condition>`
	`b`	Set a breakpoint with optional condition at a location or function

`info breakpoints`	`i b`	List all breakpoints
`delete <breakpoint number>`	`d`	Delete a breakpoint
`continue`	`c`	Resume execution

Admin

`run <args>`	`r`	Run local program
`quit`	`q`	Exit GDB
`help <cmd>`	`h`	Print quick reference for a command
`set history save`		Save command history

Thanks!
Questions?

See office hour calendar on the website for availability.

Contact:
Email: hacker@unc.edu
Twitter: @JJBakita
Web: https://cs.unc.edu/~jbakita

Old Well, University of North Carolina at Chapel Hill, Winter 2017