Lab 5

Main Goals

- **Analyze** potential return-to-libc vulnerabilities in source code
- **Exploit** these vulnerabilities in different scenarios
- **Gain** a deeper understanding of the function call convention

Task 1: Inspect the Program

- Analyze the provided source code
- Determine the potential ret2libc vulnerability
- Understand the stack layout during a function call

Task 2: Using `system` function

Four subtasks

- Subtask 1:
  - Program should not exit gracefully
  - /bin/sh is an environment variable
  - ASLR is disabled

- Subtask 2:
  - Program **should exit** gracefully
  - /bin/sh is an environment variable
  - ASLR is disabled

Task 2: Using `system` function

Four subtasks [Cont'd]

- Subtask 3:
  - Program **should exit** gracefully
  - /bin/sh **is not** an environment variable
  - ASLR is disabled

- Subtask 4:
  - Program **should exit** gracefully
  - /bin/sh **is not** an environment variable
  - ASLR is **enabled**

Note: gdb disables randomization

Task 3: Using `execl` function

- Implement this chain: `printf` → `execl` → `exit`
- With proper inputs and return addresses!

Recall `execve`:

```
int execve(char *file, char *argv[], char *env[])
```

`execl`:

variable # args

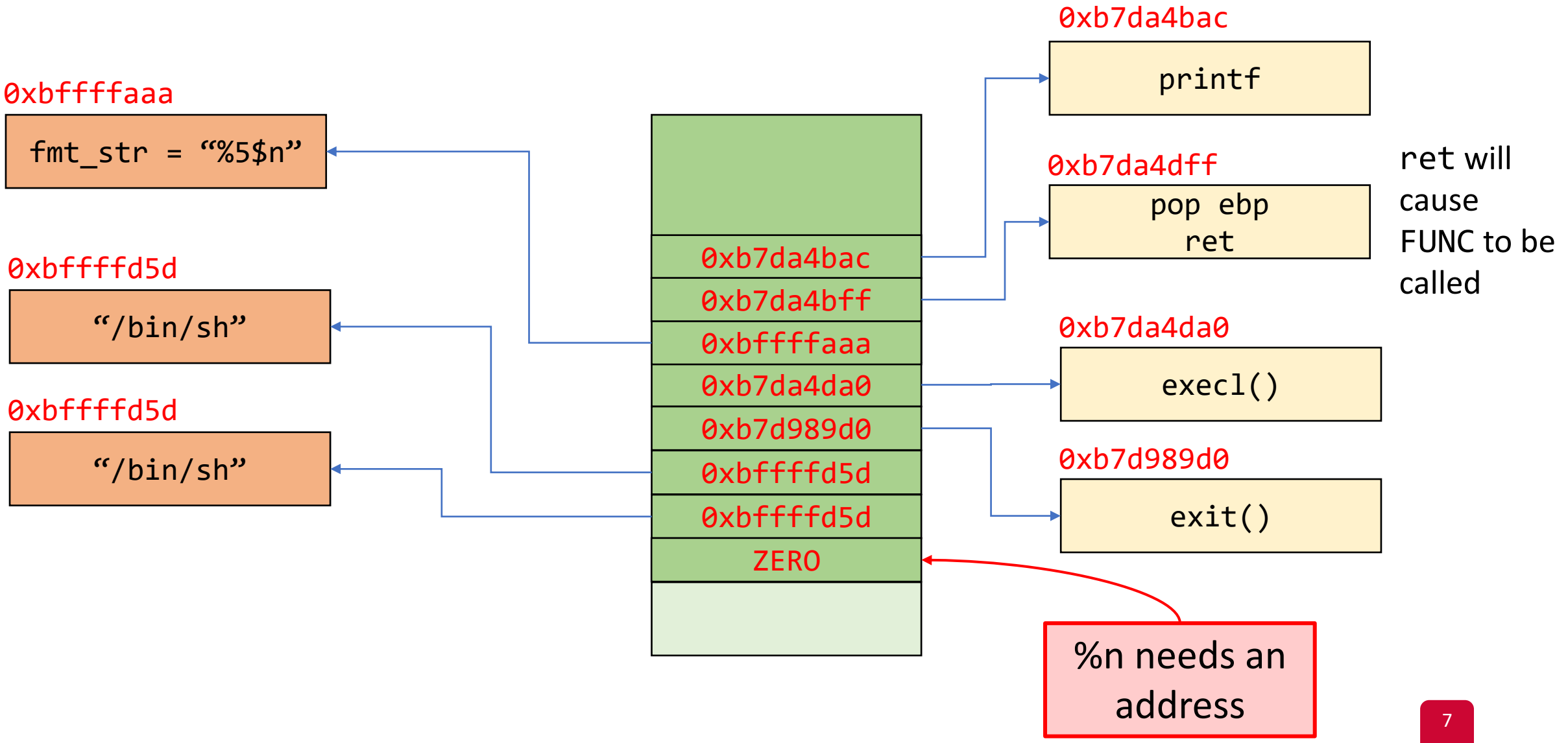
```
int execl(char *file, const char *arg, ...)
```

Calling `execl`:

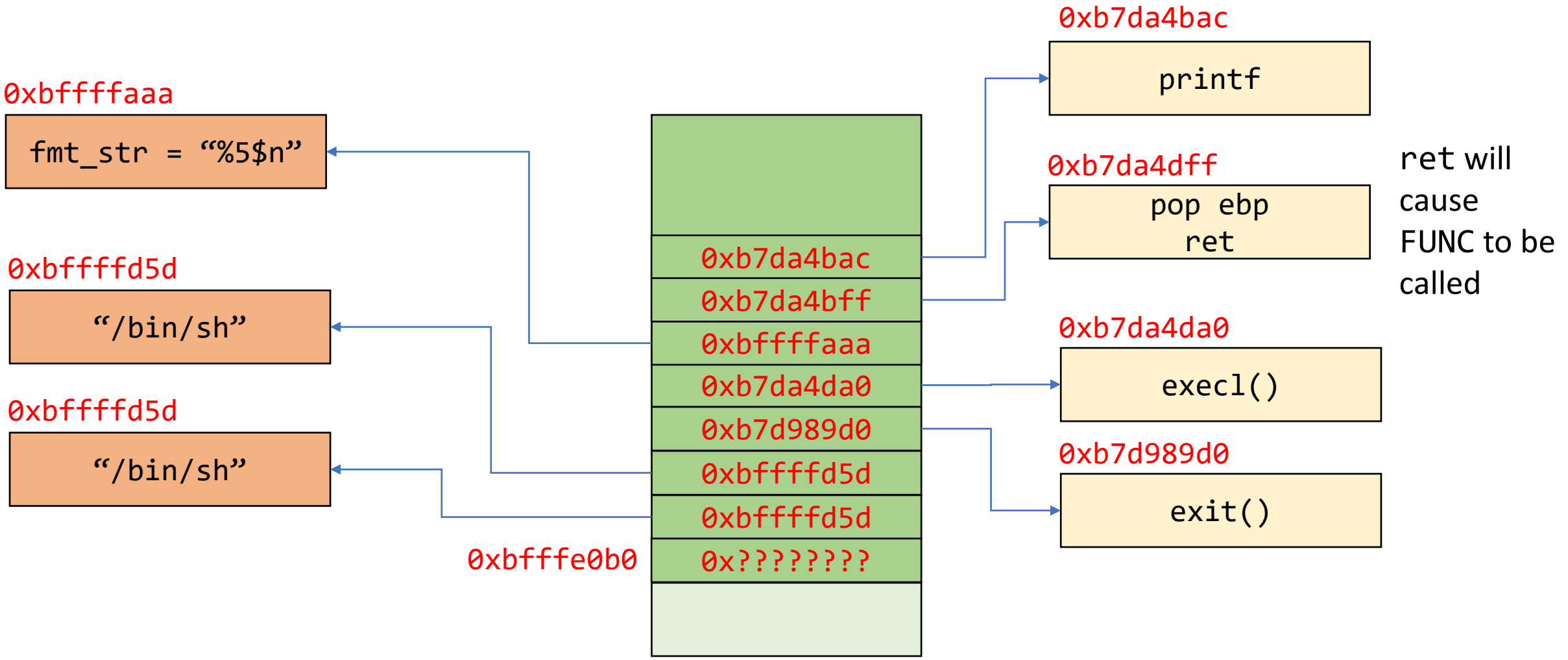
last arg is NULL

```
execl("/bin/sh", "/bin/sh", NULL);
```

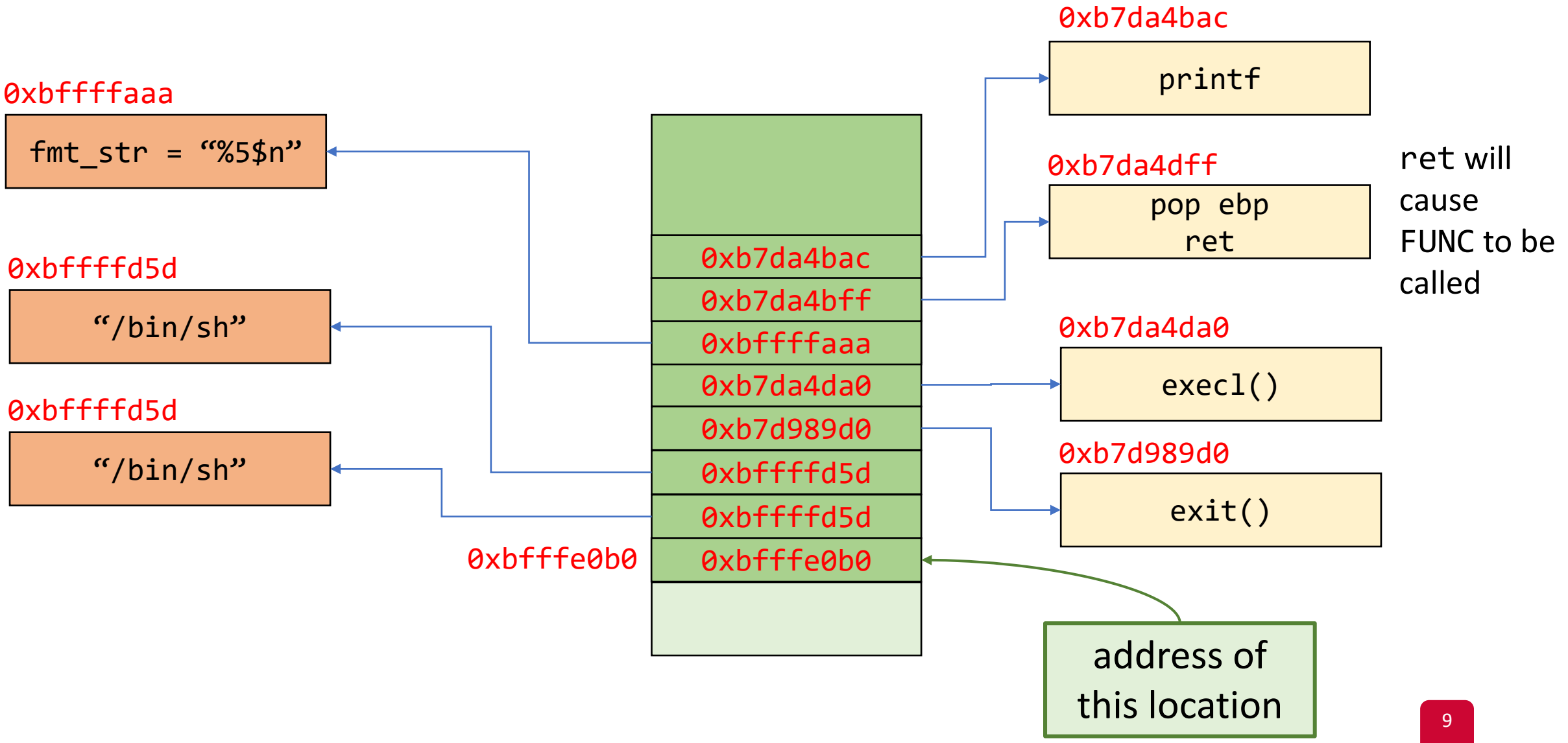
Task 3: Using `execl` function



Task 3: Using `execl` function



Task 3: Using `execl` function



Questions?