

## HW10 追蹤 execve 系統呼叫

系級：資工三 學號：409410114 姓名：周 x 君

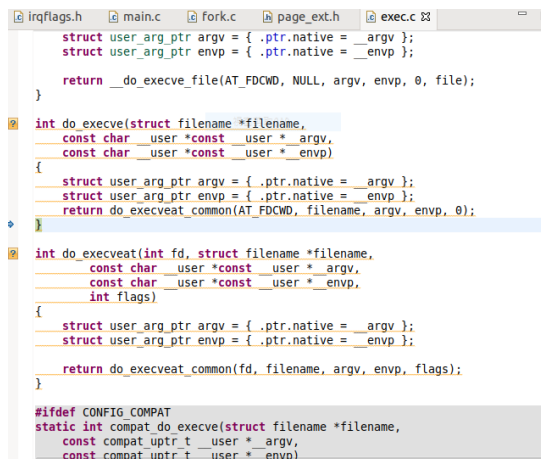
1. 撰寫程式碼稱之為 myls，在程式碼中使用 execve 系列的任何 libc 函數，載入新的執行檔案（ls）。注意：不需要使用 fork。

```
1 #include<stdio.h>
2 #include<unistd.h>
3
4 int main(){
5     int x;
6     scanf("%d", &x);
7     execl("/bin/ls", "ls", NULL);
8     return 0;
9 }
```

2. 請問作業系統如何載入執行檔案？附上程式碼的截圖，並約略說明。

`do_execve` 會將字串轉成 `struct user_arg_ptr`，接著再呼叫

`do_execveat_common`，在這邊會直接呼叫 `do_execve_file`。



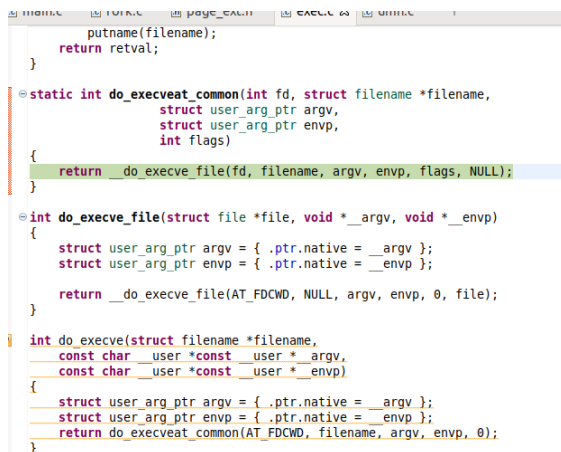
```
struct user_arg_ptr argv = { .ptr.native = __argv };
struct user_arg_ptr envp = { .ptr.native = __envp };
return _do_execve_file(AT_FDCWD, NULL, argv, envp, 0, file);

int do_execve(struct filename *filename,
const char __user *const __user * __argv,
const char __user *const __user * __envp)
{
    struct user_arg_ptr argv = { .ptr.native = __argv };
    struct user_arg_ptr envp = { .ptr.native = __envp };
    return do_execveat_common(AT_FDCWD, filename, argv, envp, 0);
}

int do_execveat(int fd, struct filename *filename,
const char __user *const __user * __argv,
const char __user *const __user * __envp,
int flags)
{
    struct user_arg_ptr argv = { .ptr.native = __argv };
    struct user_arg_ptr envp = { .ptr.native = __envp };
    return do_execveat_common(fd, filename, argv, envp, flags);
}

#ifdef CONFIG_COMPAT
static int compat_do_execve(struct filename *filename,
const compat_uptr_t __user * __argv,
const compat_uptr_t __user * __envp)
{
    struct user_arg_ptr argv = { .ptr.native = __argv };
    struct user_arg_ptr envp = { .ptr.native = __envp };
    return do_execveat_common(AT_FDCWD, filename, argv, envp, 0);
}
```

`_do_execve_file` 會準備配置記憶體(bprm)，且呼叫 `prepare_binprm`。



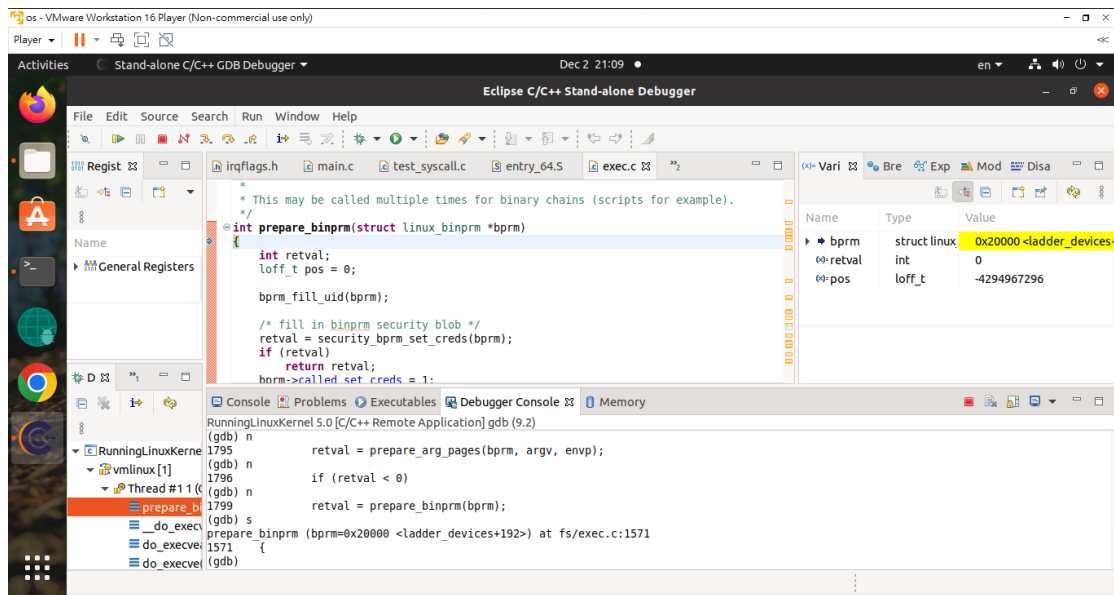
```
putname(filename);
return retval;
}

static int do_execveat_common(int fd, struct filename *filename,
struct user_arg_ptr argv,
struct user_arg_ptr envp,
int flags)
{
    return _do_execve_file(fd, filename, argv, envp, flags, NULL);
}

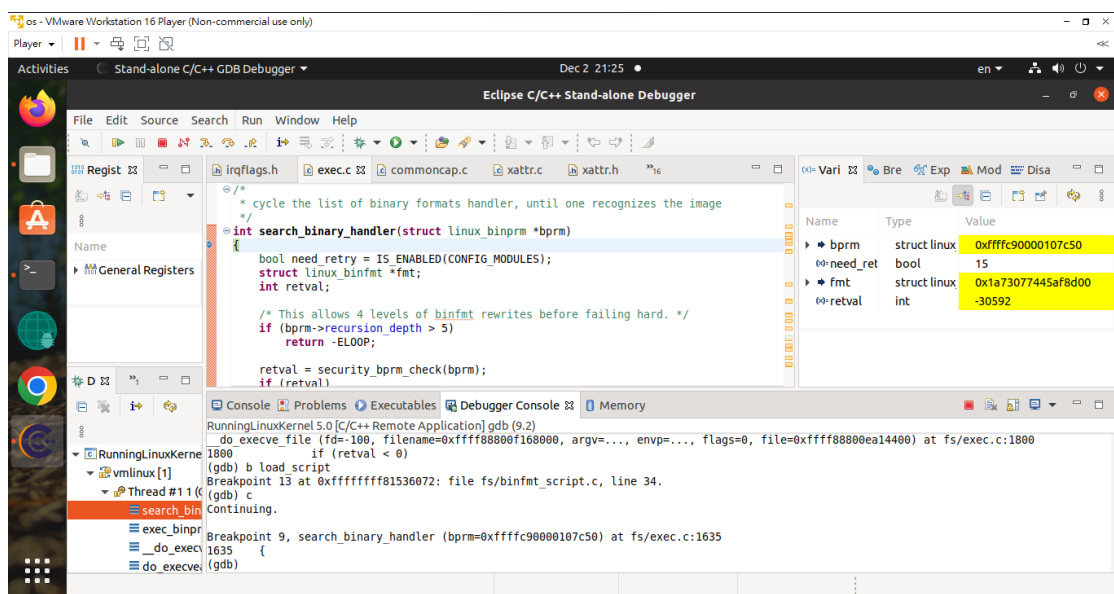
int do_execve_file(struct file *file, void * __argv, void * __envp)
{
    struct user_arg_ptr argv = { .ptr.native = __argv };
    struct user_arg_ptr envp = { .ptr.native = __envp };
    return _do_execve_file(AT_FDCWD, NULL, argv, envp, 0, file);
}

int do_execve(struct filename *filename,
const char __user *const __user * __argv,
const char __user *const __user * __envp)
{
    struct user_arg_ptr argv = { .ptr.native = __argv };
    struct user_arg_ptr envp = { .ptr.native = __envp };
    return do_execveat_common(AT_FDCWD, filename, argv, envp, 0);
}
```

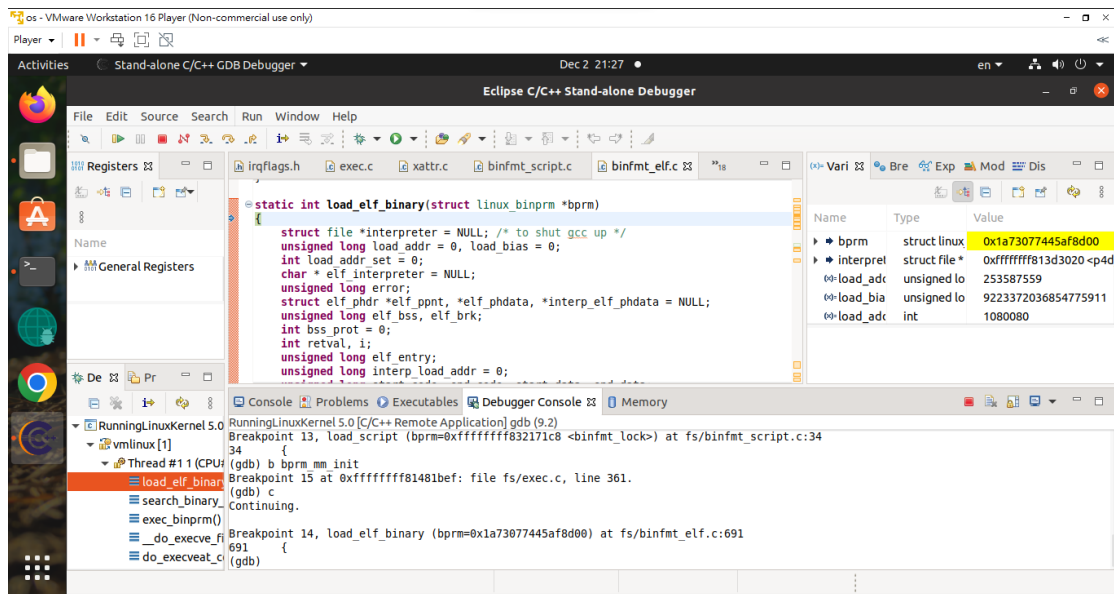
`prepare_binprm` 會將 `bprm` 的結構正式設定好，準備好後會呼叫 `retval = exec_binprm(bprm);` 執行載好的 `bprm` 文件。



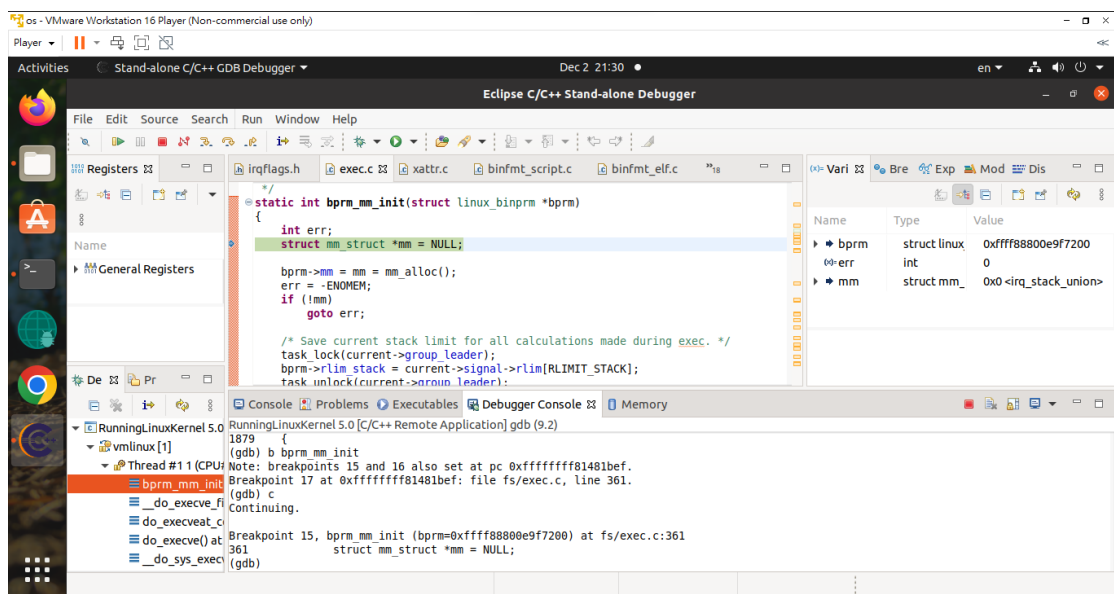
`exec_binprm` 裡會呼叫 `search_binary_handler`，函式內 `list_for_each_entry(fmt, &formats, lh)` 會跑完以 `formats` 為首的鏈結串列，搜尋所有可能執行文件的格式，找到格式後看到 `retval=fmt->load_binary(bprm)` 表示找好執行格式去下載文件了 (`bprm`)。



找到相符格式的函式是 `load_elf_binary`，用它載入新的執行檔。



3. 請問作業系統是否立即載入執行檔案到記憶體中？附上程式碼的截圖並約略說明



否，看上題執行過程可以知道，會呼叫 `_bprm_mm_init` 配置記憶體，修改 `task_struct` 中的 `mm_struct`。

參考: <https://ithelp.ithome.com.tw/articles/10186812>

ps. 因為追到一半電腦當掉換一台追，所以畫面長不太一樣。