# **Ment**oring **O**perating **S**ystem (**MentOS**)

## System call

Created by
Enrico Fraccaroli
enrico.fraccaroli@univr.it

# Table of Contents



# System Call

# System Call

## What are the ingredients in MentOs

# System Call

## Ingredients

The ingredients are:

- 1 **kernel-side** function;
- 1 **user-side** function;
- 1 **unique number** associated with the system call;

For instance:

- **kernel-side** function:

```
int sys_open(const char *pathname, int flags, mode_t mode);
```

- **user-side** function:

```
int open(const char *pathname, int flags, mode_t mode);
```

- **unique number** associated with the system call:

```
#define __NR_open 5
```

# System Call

## Folder Structure

**inc/sys/unistd.h**

- The file **defining** the **user-side** system calls;
- For instance, it contains the **open(...)** function.

**src/libc/unistd/*.c**:

- The files **implementing** the **user-side** system calls;
- Basically, they prepare the **arguments**, and call **int 80**.
- The **open(...)**, is implemented inside **src/libc/unistd/open.c**

**inc/system/syscall_types.h**

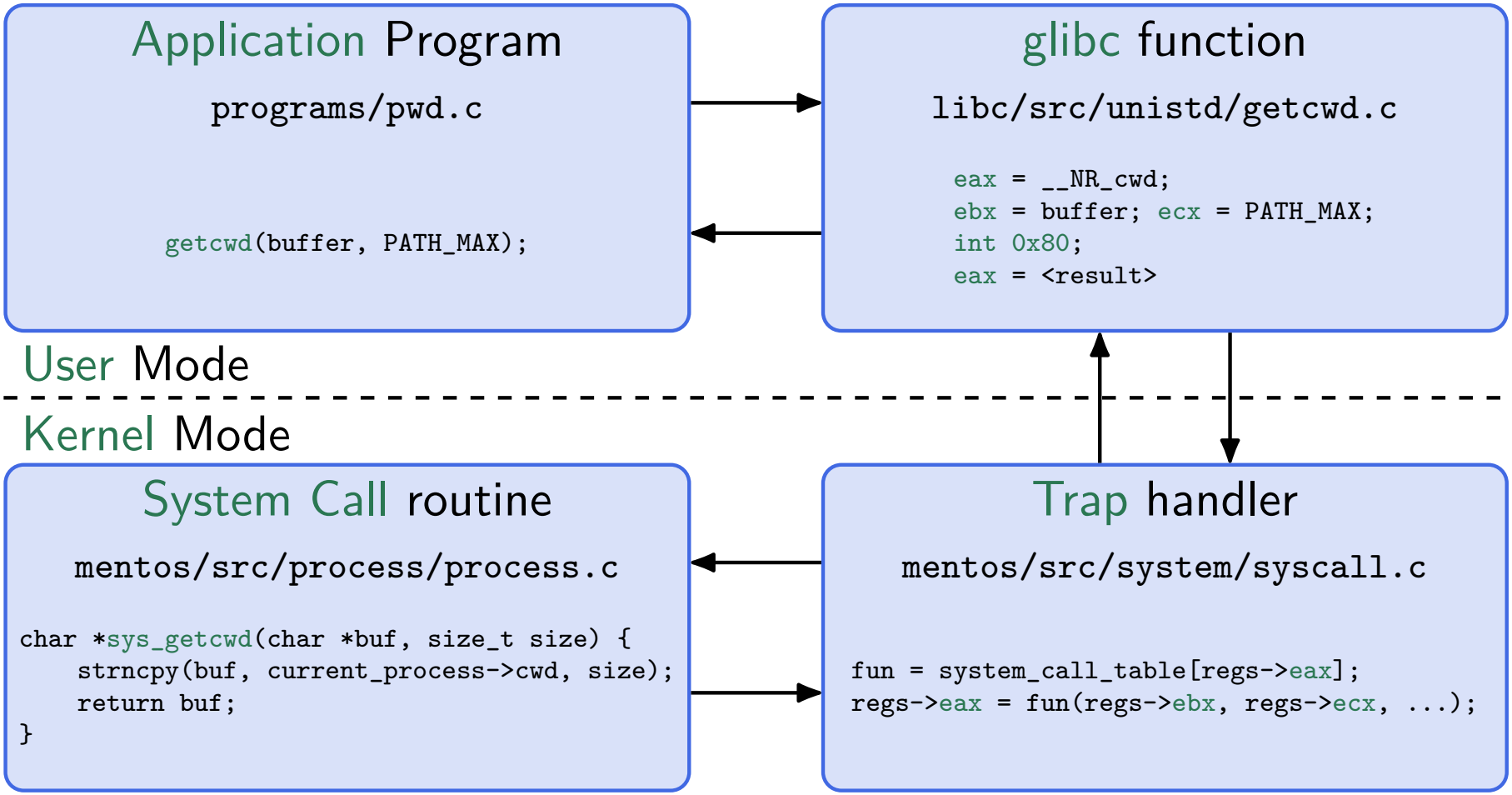
- Contains the list of **System Calls numbers**;
- The `#define __NR_open 5`;

# System Call

## How they work in MentOs

# System Call

## How they work in MentOs




# System Call

## How they work in MentOs (Example)

```
fd = open(filename, flags, mode);
```

main.c

**open(...)**

```
mov eax __NR_open
mov ebx filename
mov ecx flags
mov edx mode
int $0x80
```

src/libc/unistd/open.c

**syscall_handler(...)**

```
// The System Call number is in EAX.
sc_ptr = sc_table[regs->eax];

// Call the SC, the arguments are in EBX, EXC and EDX.
regs->eax = sc_ptr(regs->ebx, regs->ecx, regs->edx);
```

src/system/syscall.c

**sys_open(...)**

```
// Open the file with filesystem.
```

src/fs/open.c

# System Call

## Two examples of System Calls

**Preparing the registers** is done through easy-to-use **macros**:

```
int open(const char *pathname, int flags, mode_t mode) {
  ssize_t retval;
  DEFN_SYSCALL3(retval, __NR_open, pathname, flags, mode);
  if (retval < 0)
    errno = -retval, retval = -1;
  return retval;
}
```

```
int close(int fd) {
  int retval;
  DEFN_SYSCALL1(retval, __NR_close, fd);
  if (retval < 0)
    errno = -retval, retval = -1;
  return retval;
}
```