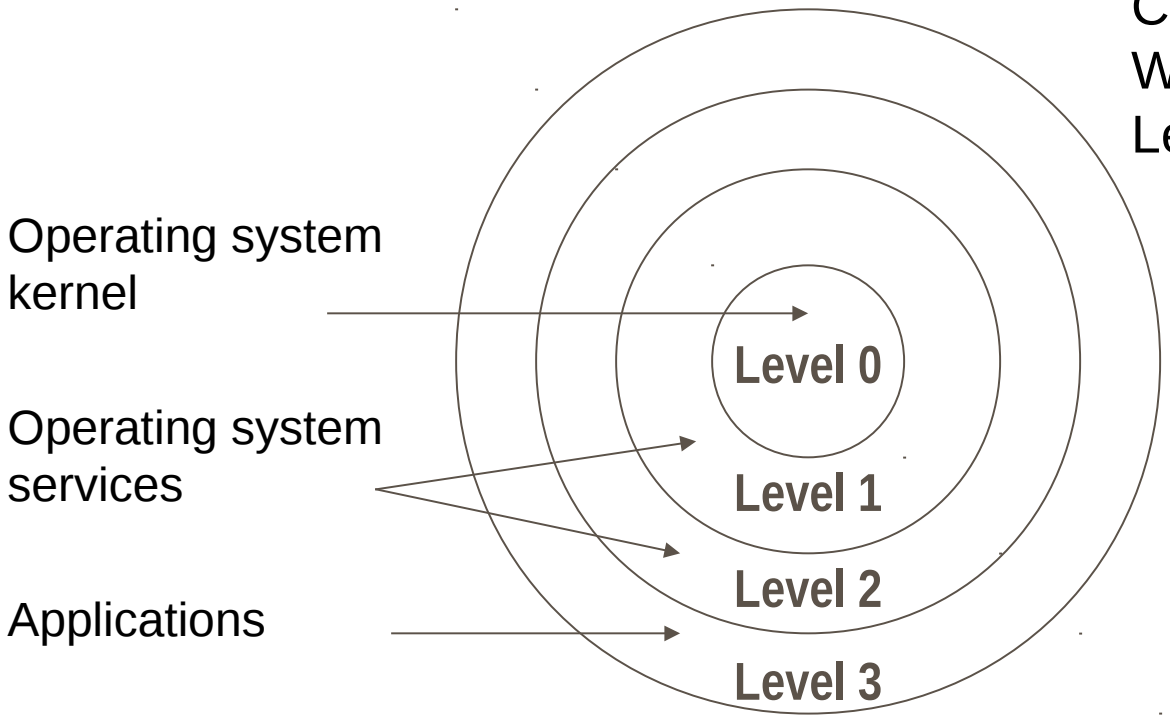
Creating a system call in Linux

Tushar B. Kute,
http://tusharkute.com

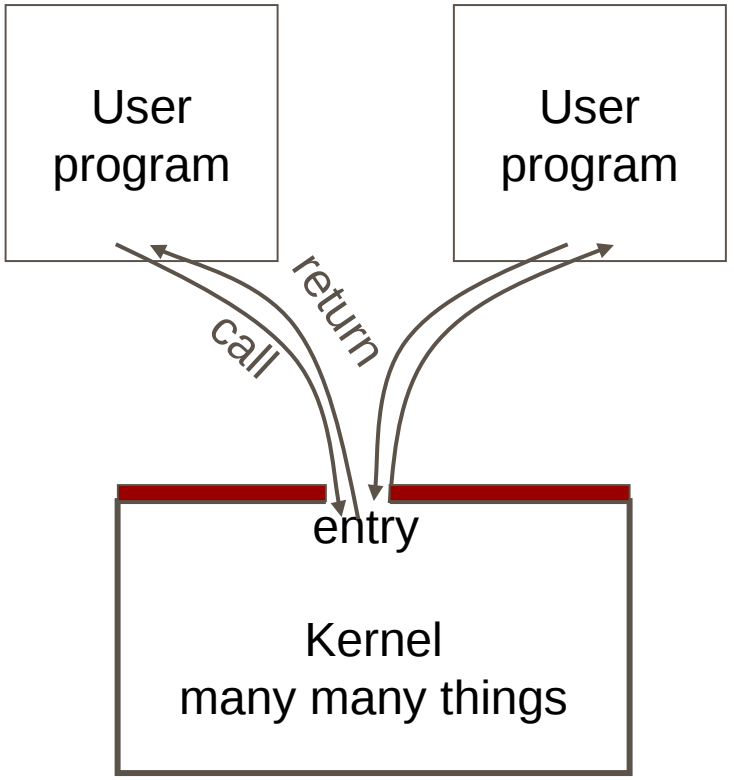
x86 Protection Rings

Privileged instructions
Can be executed only
When current privileged
Level (CPL) is 0



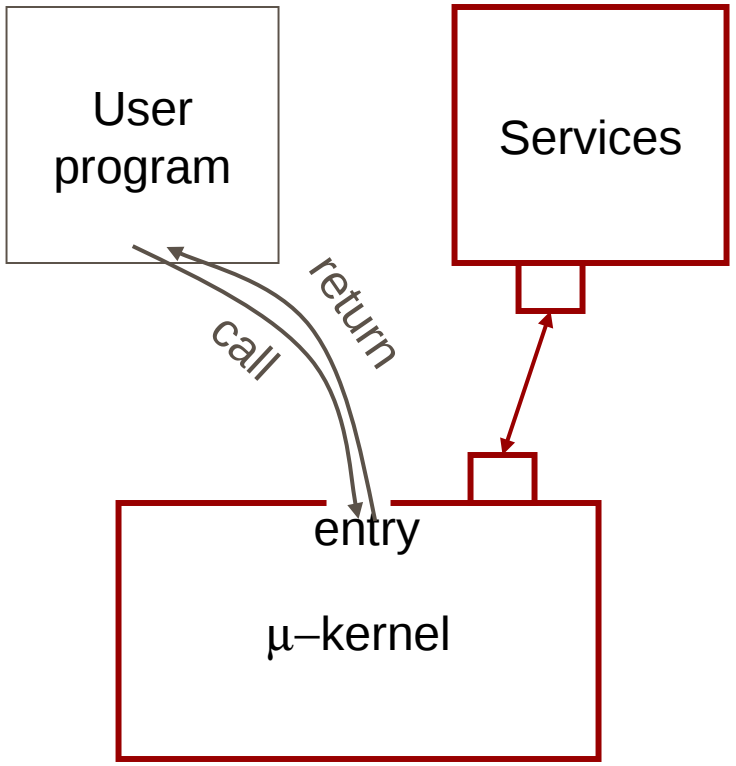
Monolithic Kernel

- All kernel routines are together.
  - A system call interface.
- Examples:
  - Linux.
  - Most Unix OS.
  - NT.



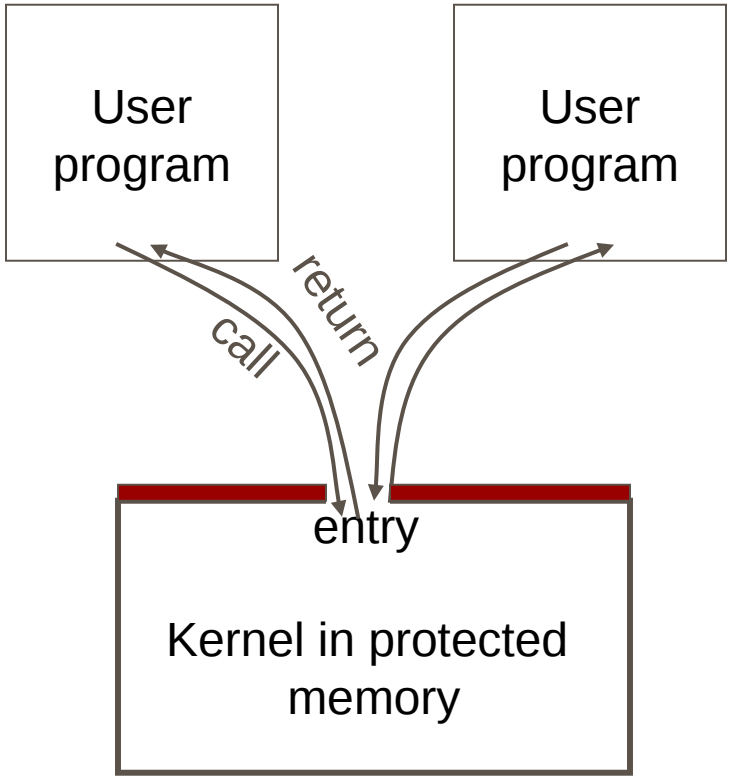
Micro Kernel

- Micro-kernel is “micro”
  - Services are implemented as regular process.
  - Micro-kernel get services on behalf of users by messaging with the service processes.
  - Examples: Taos, Mach, L4.

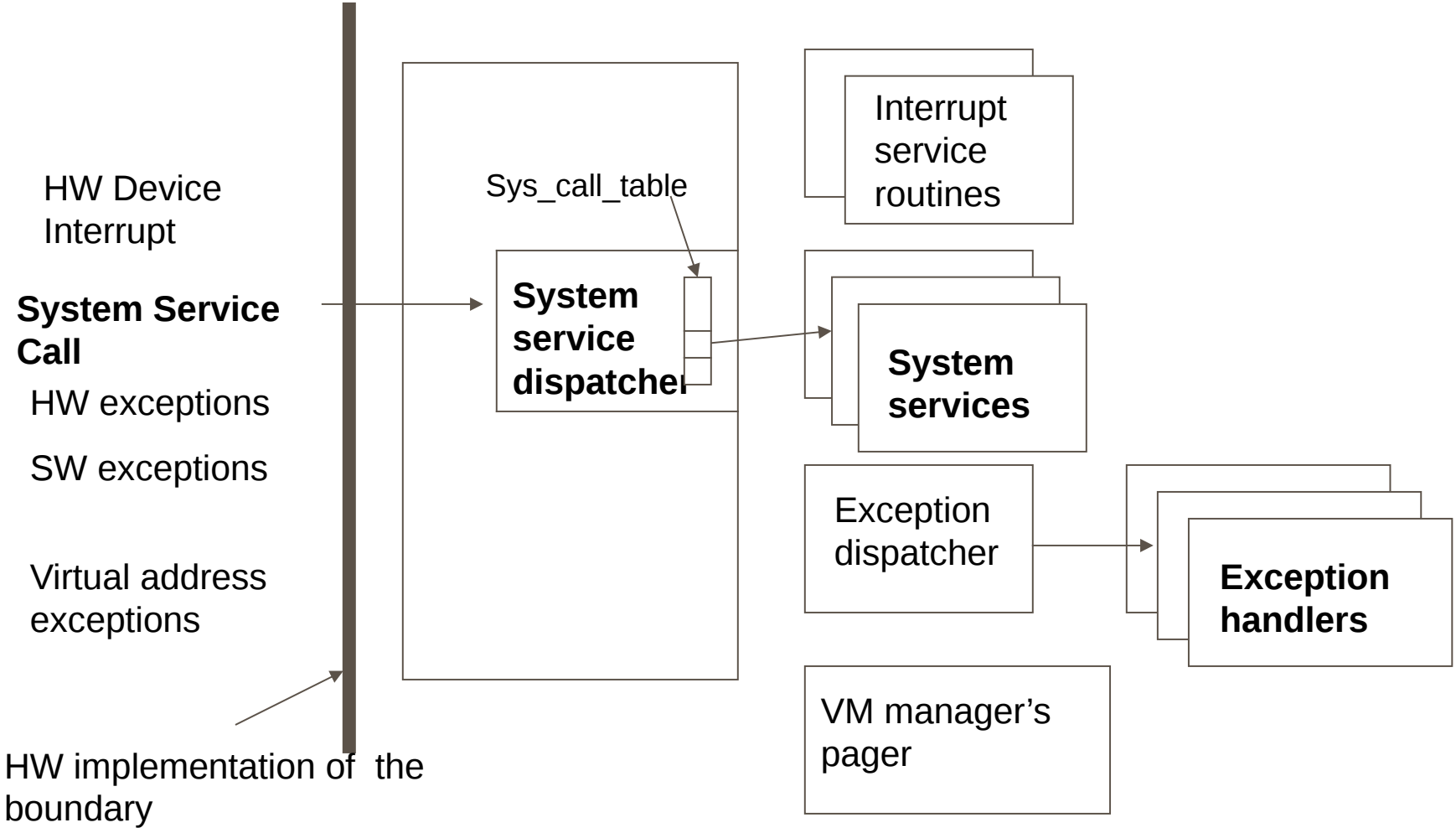


System call mechanism

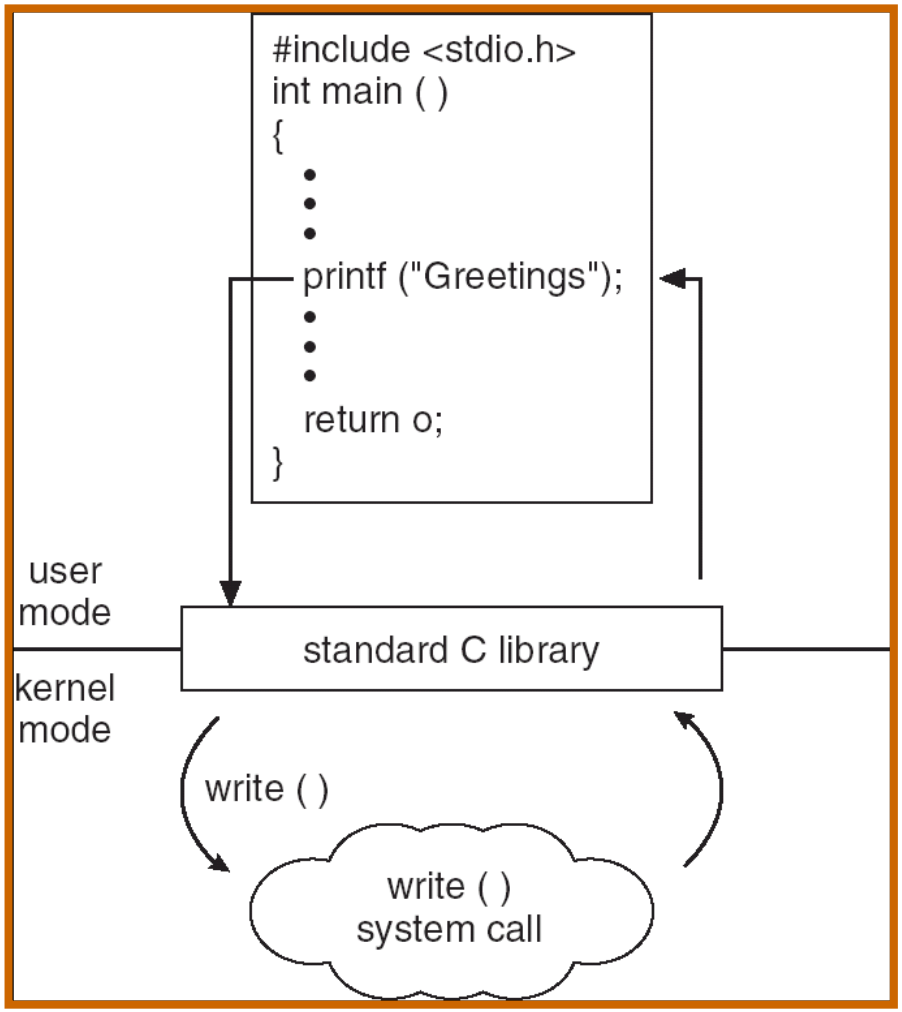
- User code can be arbitrary.
- User code cannot modify kernel memory.
- Makes a system call with parameters.
- The call mechanism switches code to kernel mode.
- Execute system call.
- Return with results.



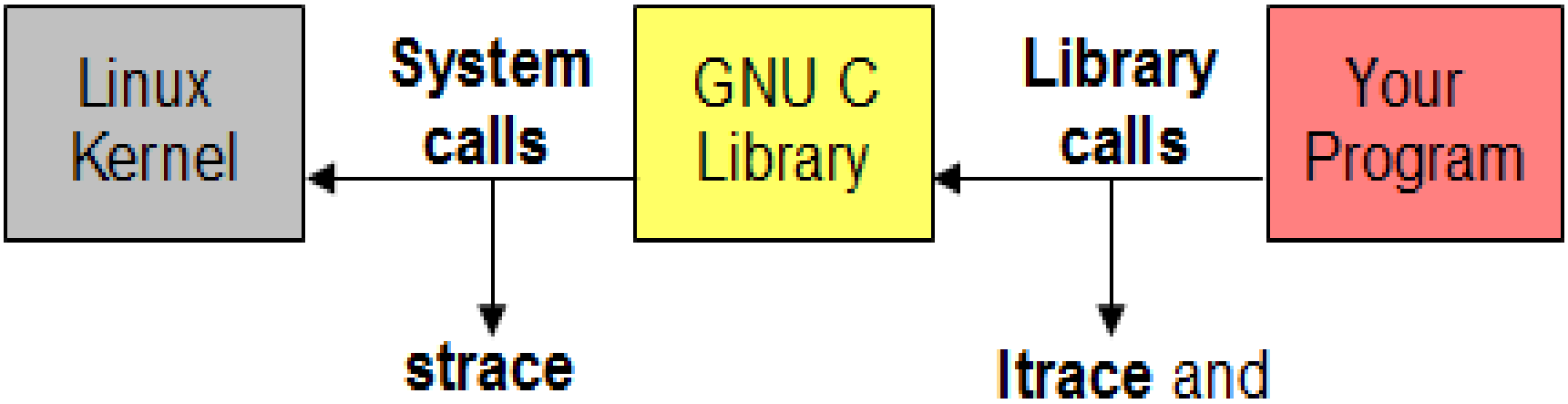
OS Kernel : Trap Handler



Library function vs. System Call



strace and ltrace



strace

- Strace monitors the system calls and signals of a specific program.
- It is helpful when you do not have the source code and would like to debug the execution of a program.
- strace provides you the execution sequence of a binary from start to end.
  - strace ls

ltrace

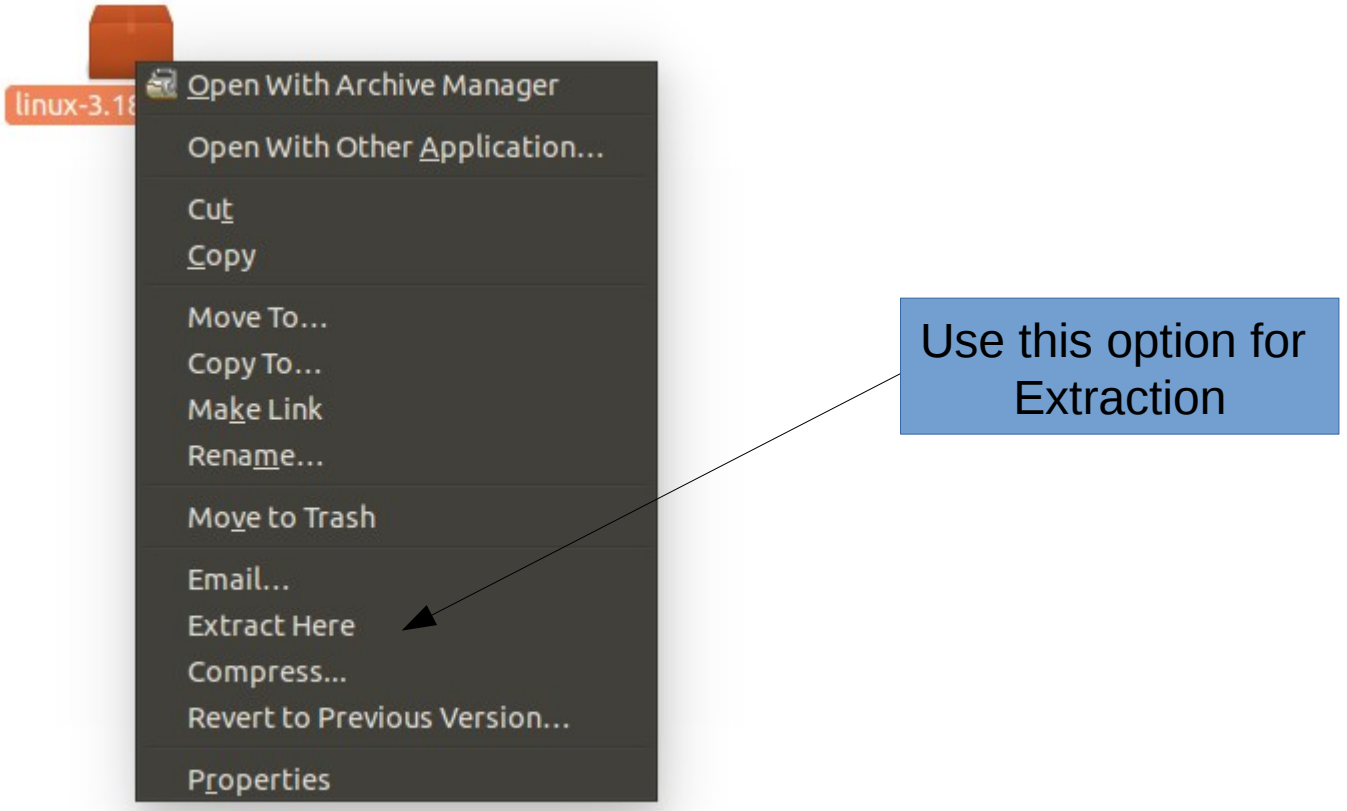
- ltrace monitors the library calls and signals of a specific program.
- It is helpful when you do not have the source code and would like to debug the execution of a program.
- ltrace provides you the execution sequence of a binary from start to end.
  - ltrace printf

Download and extract Linux source code

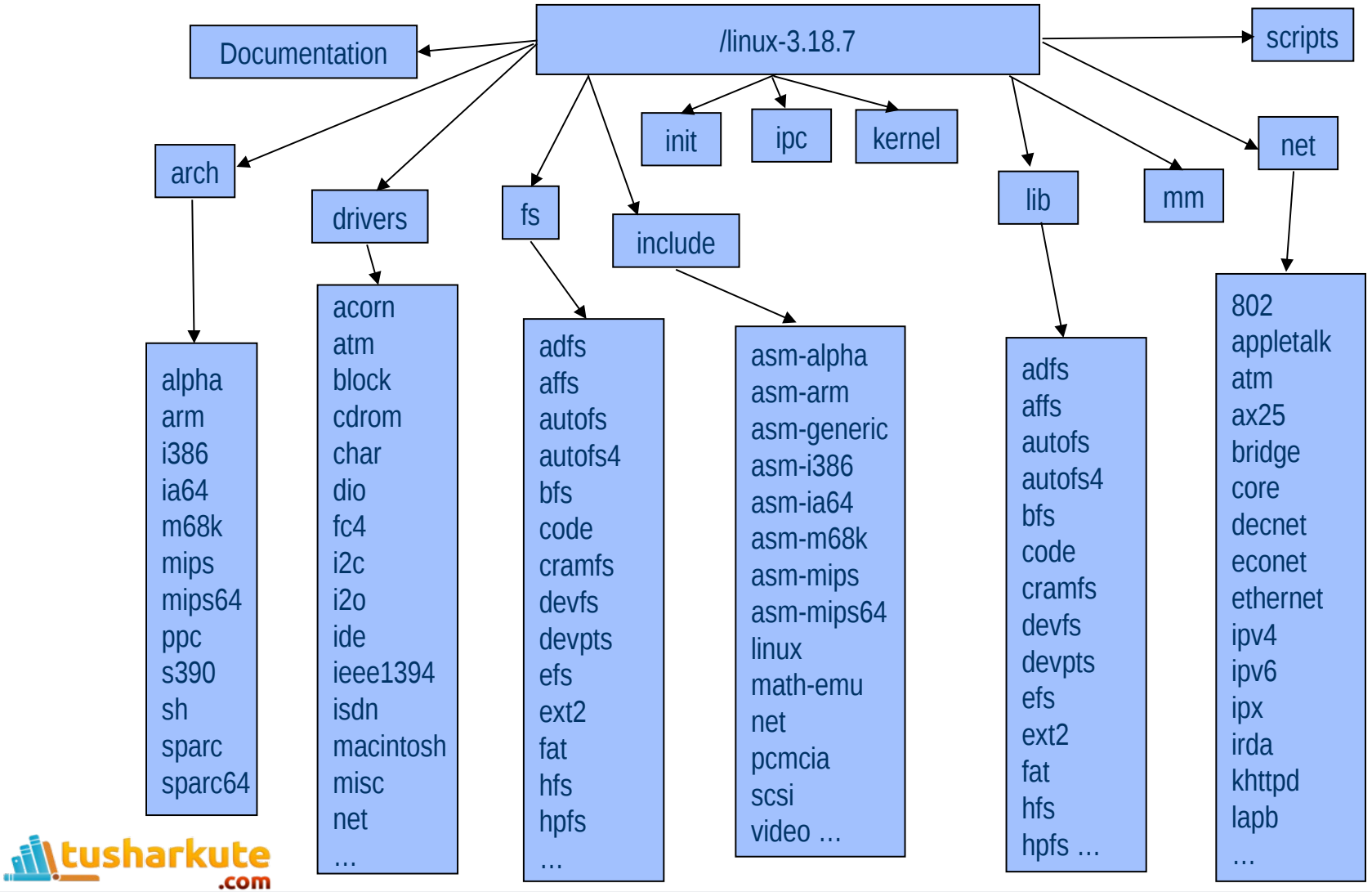
- Download the kernel source code from http://kernel.org
  - Actual link:
    https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.18.1.tar.xz
  - Can be downloaded by command:
    wget https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.18.1.tar.xz
  - Downloaded file is: linux-3.18.1.tar.xz

- Extract the file using command or GUI:
  - `tar -xvf linux-3.18.1.tar.xz`

Extraction using GUI



Linux Source Tree Layout



Define a new system call

- Create a directory hello in the kernel source directory:
  - **`mkdir hello`**
- Change into this directory
  - **`cd hello`**
- Create a “hello.c” file in this folder and add the definition of the system call to it as given below (you can use any text editor).

hello.c

```
#include <linux/kernel.h>
asmlinkage long sys_hello(void)
{
    printk(KERN_INFO “Hello world\n”);
    return 0;
}
```

Create Makefile

- Create a “Makefile” in the hello folder and add the given line to it.
  - **`gedit Makefile`**
- add the following line to it:-
  - **`obj-y := hello.o`**
- This is to ensure that the hello.c file is compiled and included in the kernel source code.

Add the hello directory to the kernel's Makefile

- change back into the linux-3.16 folder and open Makefile.
  - gedit Makefile
  - goto line number 842 which says :- "core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ "
  - change this to "core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ hello/"
- This is to tell the compiler that the source files of our new system call (sys_hello()) are in present in the hello directory.

Add system call to the table

- If your system is a 64 bit system you will need to alter the **syscall_64.tbl** file else **syscall_32.tbl**.
  - **`cd arch/x86/syscalls`**
  - **`gedit syscall_32.tbl`**
- Add the following line in the end of the file :-

```
358     i386     hello     sys_hello
```

358 – It is the number of the system call . It should be one plus the number of the last system call. (it was 358 in my system). This has to be noted down to make the system call in the userspace program.

Add system call to header file

- **cd include/linux/**
- **gedit syscalls.h**
  - add the following line to the end of the file just before the #endif statement at the very bottom.
  - **asmlinkage long sys_hello(void);**
- This defines the prototype of the function of our system call.”**asmlinkage**” is a key word used to indicate that all parameters of the function would be available on the stack.
- ***Now compile the linux source code according to the standard procedure.***

Test the system call

- Create a “userspace.c” program in your home folder and type in the following code :-

```
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>
int main()
{
    long int r = syscall(358);
    printf(“System call sys_hello returned %ld\n”, r);
    return 0;
}
```

Test the system call

- Now compile this program using the following command.
  - **gcc userspace.c**
- If all goes well you will not have any errors else, rectify the errors. Now run the program using the following command.
  - **./a.out**
- You will see the following line getting printed in the terminal if all the steps were followed correctly.
  - **“System call sys_hello returned 0“.**
- Now to check the message of the kernel you can run the following command.
  - **dmesg**

Thank you

Web Resources
http://tusharkute.com

Blogs
http://digitallocha.blogspot.in
http://kyamputar.blogspot.in

tushar@tusharkute.com