

Name : Ramzy Izza Wardhana
NIM : 21/472698/PA/20322
Class : IUP CS B – Lab OS & CN

Activity 3.2 – LAB OS

1. Save the trace of a command `echo hello` to a file titled “`echo.log`”

[illegible]

2. Filter the “echo.log” file to find the text “hello” being mentioned. Screen capture the result

```
ramzy@ramzy-VirtualBox:~$ grep hello echo.log
execve("/usr/bin/echo", ["echo", "hello"], 0x7ffdf0fe91938 /* 54 vars */) = 0
write(1, "hello\n", 6) = 6
```

3. Explain what the system call that related to the text in question number 2 is doing based on the manual page of the system call.

```
EXECVE(2)                                Linux Programmer's Manual                                EXECVE(2)

NAME
    execve - execute program

SYNOPSIS
    #include <unistd.h>

    int execve(const char *pathname, char *const argv[],
               char *const envp[]);

DESCRIPTION
    execve() executes the program referred to by pathname. This causes the program that is currently being run by the calling process to be replaced with a new program, with newly initialized stack, heap, and (initialized and uninitialized) data segments.

    pathname must be either a binary executable, or a script starting with a line of the form:
```

```
execve("/usr/bin/echo", ["echo", "hello"], 0x7fffa73ecc88 /* 23 vars */) = 0
```

The system call “execve” is a abbreviation of execute program that corresponds to the pathname. Inside the parentheses execve will run the program through the **/usr/bin/echo** directory path and with additional parameter “hello” denoted as the content that we want to filter.

```

WRITE(2)                                     Linux Programmer's Manual                                     WRITE(2)

NAME
    write - write to a file descriptor

SYNOPSIS
    #include <unistd.h>

    ssize_t write(int fd, const void *buf, size_t count);

DESCRIPTION
    write() writes up to count bytes from the buffer starting at buf to the file referred to by the file descriptor fd.

    The number of bytes written may be less than count if, for example, there is insufficient space on the underlying physical medium, or the RLIMIT_FSIZE resource limit is encountered (see setrlimit(2)), or the call was interrupted by a signal handler after having written less than count bytes. (See also pipe(7).)

    For a seekable file (i.e., one to which lseek(2) may be applied, for example, a regular file) writing takes place at the file offset, and the file offset is incremented by the number of bytes actually written. If the file was open(2)ed with O_APPEND, the file offset is first set to the end of the file before writing. The adjustment of the file offset and the write operation are performed as an atomic step.

    POSIX requires that a read(2) that can be proved to occur after a write() has returned will return the new data. Note that not all filesystems are POSIX conforming.

    According to POSIX.1, if count is greater than SSIZE_MAX, the result is implementation-defined; see NOTES for the upper limit on Linux.

```

write(1, “hello\n”, 6)

`write()` system call has the function to write data to the file descriptor with respect to count of bytes buffer starting from `buf` to the file referred by `fd`. `Write()` has accepts 3 parameters namely `fd`, `buf`, and `count`. `Fd` is the file descriptor, `buf` is the placeholder of the data, and `count` is the bytes that needs to be write. This system calls will further explain by the output given above from *man 2 write*.