```
.Dd 10/7/2023               \" DATE
.Dt Project_1      \" Program name and manual section number
.Os Reptilian
.Sh NAME                 \" Section Header - required - don't modify
.Nm Project 1: System Calls
.Sh SYNOPSIS             \" Section Header - required - don't modify
.Nm

The goal of this assignment is to implement custom system calls within
Reptilian for 3 library functions. These functions are get and set the log
level and submitting log entries!
.Sh DESCRIPTION          \" Section Header - required - don't modify
.Nm
The following describe the steps, comands and created/changed files.
.Pp                      \" Inserts a space
.Sh FILES:
.Pp
FOR ALL FILES:


During discussion and through the sources provided, I was able to
understand these file needs to be altered to create the system call entry,
prototyping and defining.

Each of the following changes/creations were directed of me by the
assingment specifications.
To see the exact changes, refer to p1.diff.
SOURCES: All sources and what was derived from it are listed at the end of
this document.


/usr/rep/src/reptilian-kernel/arch/x86/entry/syscalls/syscall_64.tbl


I added three new system call entries (435, 436, 437) for the functions
get_log_lvl, set_log_lvl, and log_msg, as per the assignment requirements.
These entries serve as entry points for the new system calls.


/usr/rep/src/reptilian-kernel/include/linux/syscalls.h


I defined the prototypes for the newly added system calls, specifying
their respective parameters. get_log_lvl has no parameters, set_log_lvl
takes an integer parameter, and log_msg takes a char* and an integer
parameter.


/usr/rep/src/reptilian-kernel/kernel/sys.c
```

I implemented the actual system call functions. I created a global integer variable named log_lvl and initialized it to 0. The functions are as follows:

get_log_lvl: Returns the global variable.

set_log_lvl: Checks is user has root access, if input is valid (0-7) and sets the global variable to the input on success, returns -1 on failure.

log_msg: Converted user space message to kernel space of max length 128 characters. Checks if log level is valid (0-7 and less than or equal to current log level). Retruns equivalent process log level message using executable and process ID number on success, -1 on failre.

/home/reptilian/process_log

Created directory to house library, contains the following files:

/home/reptilian/process_log.h

Created all prototypes for each library and harness functions.

/home/reptilian/process_log.c

Defined all library and harness functions as required, calling the system calls created beforehand.

/home/reptilian/Makefile

Runs the commands provided in the Ex1 documentation to create the .a and .o files.

/home/reptilian/testingHelp.c

Runs personal testing functions for each syscall (see TESTING)

/usr/rep/src/reptilian-kernel/p1.diff

This is the patch file required by the assingment, it contains all changes to the kernel (sycalls_64.tbl, syscalls.h, sys.c)

.Sh COMMANDS:

gcc -o file_name file_name.c -L./process_log -lprocess_log

g++ -std=c++17 -o file_name file_name.cpp -L./process_log -lprocess_log

These commands were provided in the documentation.

These commands compile the test tiles provided. The first for .c files and the second for .cpp

I used these commands because it is needed to compile and build the tests.

tar -zcvf process_log.tar.gz process_log

This command was from EX1, and it archives and zips the directory/ its elements as required by specifications.

git add -u

git add '*.c' '*.h' '*Makefile*' '*.tbl'

git diff remotes/origin/os-latest > p1.diff

Command provided from P0, it creates a patch of all changes to the kernel. I used it as directed by assingment.

.Sh TESTING
    For testing I ran the provided test files. Ran peer test files as well as my own for edge cases:

    - All Invalid Inputs

    - Over and exactly 128 character inputs

    - Invalid memory adresses

    * Done for every log level/syscall*
.Pp
.Sh BUGS
    No known bugs
.Sh LINK
    https://youtu.be/mmEdQ7C7d-A
.Sh REFERENCES
System Call Paths:

https://www.cise.ufl.edu/research/reptilian/wiki/doku.php?id=modifying_the_linux_kernel

https://williamthegrey.wordpress.com/2014/05/18/add-your-own-system-calls-to-the-linux-kernel/

Copy_from_user documentation:
https://archive.kernel.org/oldlinux/htmldocs/kernel-api/API---copy-from-user.html

Getting PID (current->pid) and Executable(current->comm):

https://stackoverflow.com/questions/22346545/current-in-linux-kernel-code

Makefile:

https://cs.colby.edu/maxwell/courses/tutorials/maketutor/

Man File:

https://roperzh.github.io/grapse/

General Aid for syntax & implementation (harness functions etc.):
https://chat.openai.com/

.Sh AUTHOR

Ivan Saldarriaga