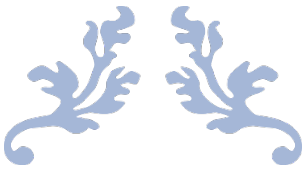

BLG 413E
Project 1 Report

Group Members

ID	Name	Surname
150120138	Emre	Özdil
150140115	Arzugül Merve	Ecevit

1) INTRODUCTION

For this project, we wrote a system call which sets the value of myFlag in the task descriptor of a process and modified the exit system call to change its behaviour based on the value of myFlag.

System calls are made for sending a request to kernel of the system. To create or execute a process, system calls are used.

The value of myFlag can be 1 or 0. set_myFlag system call sets the value of myFlag. In system call, for the given cases, errors will be returned:

- If process does not have root privileges
- If process with the given Pid cannot be found
- If the value of myFlag is different from 0 or 1

myFlag is used in exit system call. If myFlag is equals to 0, in exit system call everything will work as the same. However, if myFlag is equals to 1, in exit system call if the nice value of process is greater than 10, all children of the exiting process will be terminated along with the process itself.

2) CHANGED FILES

a) /include/linux/sched.h

We added the following line to the end of task descriptor. Since task descriptor works with pointers, this line should be added to the end.

```
int myFlag;
```

b) /include/linux/init_task.h

We added the following line to init_task.h to initialize the myFlag for all processes.

```
.myFlag     = 0
```

c) /kernel/fork.c

We added the following line to fork.c. Because, if a child is created it should get the myFlag of its parent.

```
p->myFlag = 0;
```

d) /set_myFlag

We created the system call as the following. System call detects the errors which are descriped in introduction part.

```
asmlinkage long set_myFlag(pid_t pid, int value) {

	if ((current->cred)->uid == 0) {
		struct task_struct *tsk;
		tsk = find_task_by_vpid(pid);
		if (tsk != NULL) {
			if (value == 0 || value == 1){
				tsk->myFlag = value;
			} else {
				return -EBADMSG;
			}
		} else {
			return -ESRCH;
		}
	} else {
		return -EACCES;
	}
	return 0;
}
```

e) /arch/x86/syscalls/syscall_32.tbl

We added following line to syscall_32.tbl. This file consists of system call numbers and entry vectors. System calls should be added to here.

```
355 i386  set_myFlag    set_myFlag
```

f) /include/linux/syscalls.h

We added following line to syscalls.h. The declaration of system calls are stored in here. New system calls should be added here.

```
asmlinkage long set_myFlag(pid_t pid, int flag);
```

g) /Makefile

We added following line to Makefile. Using makefile, we ensured about only the files that have been modified since the last build and those which are dependent on the changed files are only compiled.

```
	obj-y := set_myFlag.o
```

h) /kernel/exit.c

We added following lines to exit.c. By the following lines, if nice value is bigger than 10 and myFlag equals to 1, the parent process will die along with its children.

```
	struct task_struct *tsk = current;
	int group_dead;
	struct list_head *traverse;
	struct task_struct *child;

	if(tsk->myFlag == 1 && task_nice(tsk) > 10){
		list_for_each(traverse, &current->children) {
			child = list_entry(traverse, struct task_struct, sibling);
			sys_kill(child->pid, SIGKILL);
		}
	}
	profile_task_exit(tsk);
```