Assignment UP-02

In syscall.c,

Data Structure:

```
typedef struct action{
  int argc;                        // number of argument in a function
  int (*function) ();              // is valid if function has return type int
  void (*func) ();                 // is valid if function has return type void
}action;

// {No of arguments, name if function is of return type int, name if function is of return type void}
static const action actions[]={
  {0, NULL, halt},
  {1, NULL, exit},
  {1, exec, NULL},
  {1, wait, NULL},
  {2, create, NULL},
  {1, remove, NULL},
  {1, open, NULL},
  {1, filesize, NULL},
  {3, read, NULL},
  {3, write, NULL},
  {2, NULL, seek},
  {1, tell, NULL},
  {1 ,NULL, close}
};
```

Functions modified:

void syscall_handler (struct intr_frame *f) ;

Calls functions according to system call stored in stack.

Function added

```
static void validate (const int *ptr)
        // Validates if ptr!=NULL and ptr is below PHY_BASE

static void halt()
        // calls power off function

void exit (int status)
        // Calls thread_exit

static int exec (const char *cmd_line)
        // Not implemented yet

static int wait (int pid)
        // Not implemented yet

static int create (const char *file, unsigned initial_size)
        // Not implemented yet

static int remove (const char *file)
        // Not implemented yet

static int open (const char *file)
        // Not implemented yet

static int filesize (int fd)
        // Not implemented yet

static int read (int fd, void *buffer, unsigned size)
        // Not implemented yet

static int write (int fd, const void *buffer, unsigned size)
        // writes from buffer to console if fd == 1.

static void seek (int fd, unsigned position)
        // Not implemented yet

static int tell (int fd)
        // Not implemented yet
```

static void close (int fd)
    // Not implemented yet

Algorithm:

1. Made a struct actions to call functions according to the system calls given in stack.
2. Validates if stack pointer is valid or not by calling function validate().
3. Then the function is called after checking its return type and no of arguments.
4. Wrote the functions halt(), exit() and write() as given in PINTDOC.

In process.c

Function modified:

Int process_wait (tid_t child_tid)
    // Parent wait till child terminate

Algorithm:

1. Check the status of the child thread given the tid of child thread.
2. If status is not THREAD_DYING, it calls thead_yield until status becomes THREAD_DYING.

In thread.c

Function added

bool check_child_status(tid_t child_tid)
    // Given the tid , it check if thread status is THREAD_DYING

Algorithm:

Iterate over all_list and find the thread with the given tid and return true if thread status is not equal to THREAD_DYING and else returns false.