CS326 Operating System
Project 2: User Program
Design Document

Kaiming Yang, Hao Chen & Wanzhang Sheng

October 2014

1 Argument Parsing

void process_execute(const char* cmd) split the command line and pass the argument to the newly created process.

void start_process(void* arg_) accept the argument, initialize thread struct, load the program and jump to run.

struct process_init_arg stores the argument for initializing a thread, include parent process, semaphore for synchronize, arguments

When tries to start a process, *process_execute* will be called with the program name and the arguments as a whole string. Inside the *process_execute*, program will alloc a new page for arguments, copy the command string to the newly created page, split the string (by replace all spaces to '\0'), then pass the argument page to a newly created thread. Then *process_execute* will wait for new process finish loading.

After the new thread get the arguments, it will set the parenthood(which will be discussed in next chapter), load the program, notify his parent with the result of loading, then set up the stack and jump to user code.

2 Process Waiting and Exiting

int process_wait(tid_t pid) wait for a child process.

void process_exit() release the resource of the process, this method will be called no matter the process is exiting normally or terminated by kernel.

The parenthood is added by some field of *struct thread*(i.e. *struct thread* parent* and *struct list children*) at the process creating time. When the thread is finished or terminited, it will till its parent that return code is ready by increment a semaphore *sema_exit* and will be blocked until parent notify it that

the return code has been received by decrease a semaphore *sema_exit_ack*, and also this thread will increase *sema_exit_act* for all of its children so that they will not be blocked when they are exiting.

When a process tries to wait a child, it will decrease the *sema_exit* of the child to wait child exiting, then store the return code, and increase *sema_exit_ack* to tell the child that the result is received and you are clear to continue exiting.

3 IO Operation

struct process_file_record is list element that stores the ownership of a file descripter.

struct lock filesys_lock is the lock to synchronize file system operation.

The executable file will be keep opened while it is running and after load, the file will deny any write until the program exit. When the process is exiting, all files it opened will be closed.