# **Addis Ababa University**

# **Addis Ababa Institute of Technology**

# **Operating Systems**

## **LAB 02**

Objective: Familiarize with process and threads

* Process creation/operations
* Threads

# **Process Creation**

**Exercise 1. Create a simple process**

|  |  |  |
| --- | --- | --- |
| #include <stdio.h>  int main()  {  fprintf(stderr, "Please wait ");  int i;  for(i=0;i<10;i++)  {  fprintf(stderr, ". ");  sleep(10);  }  fprintf(stderr, "\nDone.\n");  } |  | * Run the code * Open a new terminal and check all running processes of the current user (**ps –u**) * or use the **(ps –al) command** * What is the process ID * What is the State of process. |

**Exercise 2. Inspecting a running process**

|  |  |  |
| --- | --- | --- |
| **#include <stdio.h>**  **int main()**  **{**  **printf("Running …. \n");**  **int fd1 = creat("A.txt",0777);**  **int fd2 = creat("B.txt",0777);**  **while(1) {}**  **}** |  | * What is the purpose of the while loop? * Run the code and check both the ID and state of the process. * Find the process ID in /proc directory **(cd /proc)** * Change to the process ID **(cd xxxx);** where xxxx is the process ID * Inspect contents of the directory; e.g. list the content of fd. * Kill the process using the kill command |

**Exercise 2. Using fork to create a child process**

|  |
| --- |
| **#include <stdio.h>**   * Run the code and inspect the state of the both the child process and parent process. * on a new terminal, continuously run **ps –al** to see the child process transitioning from **Sleep state** to **Zombie state** before it gets terminated.   **#include <unistd.h>**  **int main()**  **{**  **int ret;**  **ret = fork();**  **printf("fork() returned %d...\n", ret);**  **if (ret > 0)**  **{**  **sleep(75);**  **printf("\nParent ProcessID = %d\n", getpid());**  **}**  **if (ret == 0)**  **{**  **sleep(25);**  **printf("\nChild ProcessID = %d and its Parent ProcessID = %d)\n", getpid(),getppid());**  **}**  **return 0;**  **}** |

# **Threads**

**Exercise 1. A simple example**

|  |
| --- |
| **#include <pthread.h>**   * When compiling add –lpthread **(e.g. gcc –o a a.c –lpthread)**   **#include <stdio.h>**  **#include <stdlib.h>**  **void \*Display(void \*ThreadNumber)**  **{**  **int thread\_no = (int)ThreadNumber;**  **unsigned self\_id = (unsigned)pthread\_self();**  **printf("The ID of thread number %d is %u\n",thread\_no, self\_id);**  **}**  **int main()**  **{**  **pthread\_t THREADS[4];**  **int ret; int t;**  **for (t=0;t<4;t++)**  **{**  **printf("creating thread %d\n",t);**  **ret = pthread\_create(&THREADS[t], NULL, Display, (void\*)t);**    **if (ret)**  **{**  **printf("Error: Code = %d",ret);**  **exit (-1);**  **}**  **}**  **pthread\_exit(NULL);**  **}** |

**Exercise 2.** Write a program that adds numbers 1 to 32M using 4 threads. And compare its performance with a single thread implementation.