Operating Systems (ECE3325-001)

Week 6
Shell Programming
(Shell scripting)

Dept. of Electrical & Computer Engineering

Prof. Kim Deok-Hwan

Lab Assistant: Srinidhi

Environment Setup

인하대학교 INHA UNIVERSITY

- Ubuntu 20.04 LTS
  - Installation through
    - ✓ Dual boot the PC
    - ✓ VMware Workstation 16 player
    - ✓ VirtualBox

Variables: defaults

- $# → Number of variables the script was called with
- $0 → basename of the program as it was called
- $1 .. $9 → Variables in order 1 to 9
- $@ → All parameters
- $* → Similar to "$@" with slightly different output syntax

var1.sh

```
#!/bin/sh
echo "I was called with $# parameters"
echo "My name is $0"
echo "My first parameter is $1"
echo "My second parameter is $2"
echo "All parameters are $@"
```

```
sri@ubuntu:~/week6$ ./var1.sh
I was called with 0 parameters
My name is ./var1.sh
My first parameter is
My second parameter is
All parameters are
sri@ubuntu:~/week6$ ./var1.sh hello good morning
I was called with 3 parameters
My name is ./var1.sh
My first parameter is hello
My second parameter is good
All parameters are hello good morning
sri@ubuntu:~/week6$
```

Variables: IFS

인하대학교 INHA UNIVERSITY

- IFS → Internal Field Seperator

```
#!/bin/sh
old_IFS="$IFS"
IFS=:
echo "Please input some data separated by colons ..."
read x y z
IFS=$old_IFS
echo "x is $x y is $y z is $z"
```

```
sri@ubuntu:~/week6$ ./ifs.sh
Please input some data separated by colons ...
how:are:you
x is how y is are z is you
sri@ubuntu:~/week6$ ./ifs.sh
Please input some data separated by colons ...
how:are:you:doing
x is how y is are z is you:doing
```

Variables: curly brackets {var}

인하대학교 INHA UNIVERSITY

- Curly brackets around a variable avoid confusion
- It is recommended to use curly brackets if the variable is used:
  - inside the words
  - Two more variables are combined
  - Etc.

```
foo=sun
echo $fooshine      # $fooshine is undefined
echo ${foo}shine    # displays the word "sunshine"
```

Variables: defaults

- `whoami` assigns the variable to the current system username
- :- assigns the default if the variable is unset

```
#!/bin/bash
echo -en "What is your name [ `whoami` ] "
read myname
if [ -z "$myname" ]; then
  myname=`whoami`
fi
echo "Your name is : $myname"
```

VS

```
#!/bin/bash
echo -en "What is your name [ `whoami` ] "
read myname
echo "Your name is : ${myname:-`whoami`}"
```

External programs

인하대학교 INHA UNIVERSITY

- The backtick (`) indicates that the enclosed text executes as a command

```
sri@ubuntu:~/week6$ grep "^${USER}:" /etc/passwd | cut -d: -f5
srinidhi,,,
sri@ubuntu:~/week6$ MYNAME=`grep "^${USER}:" /etc/passwd | cut -d: -f5`
sri@ubuntu:~/week6$ echo $MYNAME
srinidhi,,,
sri@ubuntu:~/week6$
```

Functions

인하대학교 INHA UNIVERSITY

- Functions allow to write a block of code to execute more than once

```
#!/bin/sh
# A simple script with a function...

add_a_user()
{
  USER=$1
  PASSWORD=$2

  echo "Adding user $USER ..."
  echo useradd $USER
  echo passwd $USER $PASSWORD
  echo "Added user $USER with pass $PASSWORD"
}

###
# Main body of script starts here
###
echo "Start of script..."
add_a_user sri123 sri
echo "End of script..."
```

Practice: Factorial in shell

인하대학교 INHA UNIVERSITY

- Program that calculates the factorial
- Copy the program
- Test
- Try different inputs

factorial.sh

```
#!/bin/sh

factorial()
{
  if [ "$1" -gt "1" ]; then
    i=`expr $1 - 1`
    j=`factorial $i`
    k=`expr $1 \* $j`
    echo $k
  else
    echo 1
  fi
}


while :
do
  echo "Enter a number:"
  read x
  factorial $x
done
```

Operating Systems (ECE3325-001)

Week 6
Thread Programming

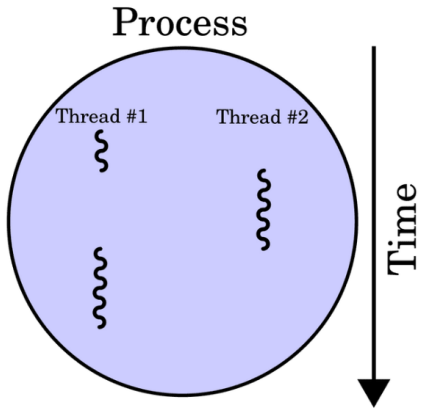
Dept. of Electrical & Computer Engineering

Prof. KIM DEOKHWAN

Lab Assistant: Srinidhi

Multithreading in C

- What is a Thread?
  - Single sequence stream with a process
  - Lightweight processes



- Process vs Thread
  - Process is independent, thread is not!
  - Shares: code, data, OS resources
  - Independent: program counter, register set, stack

- Why Multithreading?
  - Faster creation
  - Faster context switching
  - Faster inter-thread communication
  - Faster termination

✓ POSIX Threads (or Pthreads) is a POSIX standard for threads.

Example


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> //Header file for sleep().
#include <pthread.h>

// A normal C function that is executed as a thread
// when its name is specified in pthread_create()
void *myThreadFun(void *vargp)
{
        sleep(1);
        printf("Printing HELLO WORLD from Thread \n");
        return NULL;
}

int main()
{
        pthread_t thread_id;
        printf("Before Thread \n");
        pthread_create(&thread_id, NULL, myThreadFun, NULL);
        pthread_join(thread_id, NULL);
        printf("After Thread \n");
        exit(0);
}
```

Threads in C

`pthread_t` ➔ data type to uniquely identify a thread in the system

`pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start) (void *), void *arg)` ➔ function used to create a new thread

- *`thread`* ➔ address of new thread ID
- *`attr`* ➔ attributes for a new thread; Default: NULL
- *`start`* ➔ address of the function to run in a new thread
- *`arg`* ➔ arguments passed to a function

`pthread_join(pthread_t thread, void **value_ptr)` ➔ wait for thread termination

- *`thread`* ➔ ID of the thread to terminate
- *`value_ptr`* ➔ stores the value from *pthread_exit();* Default: NULL

Execution

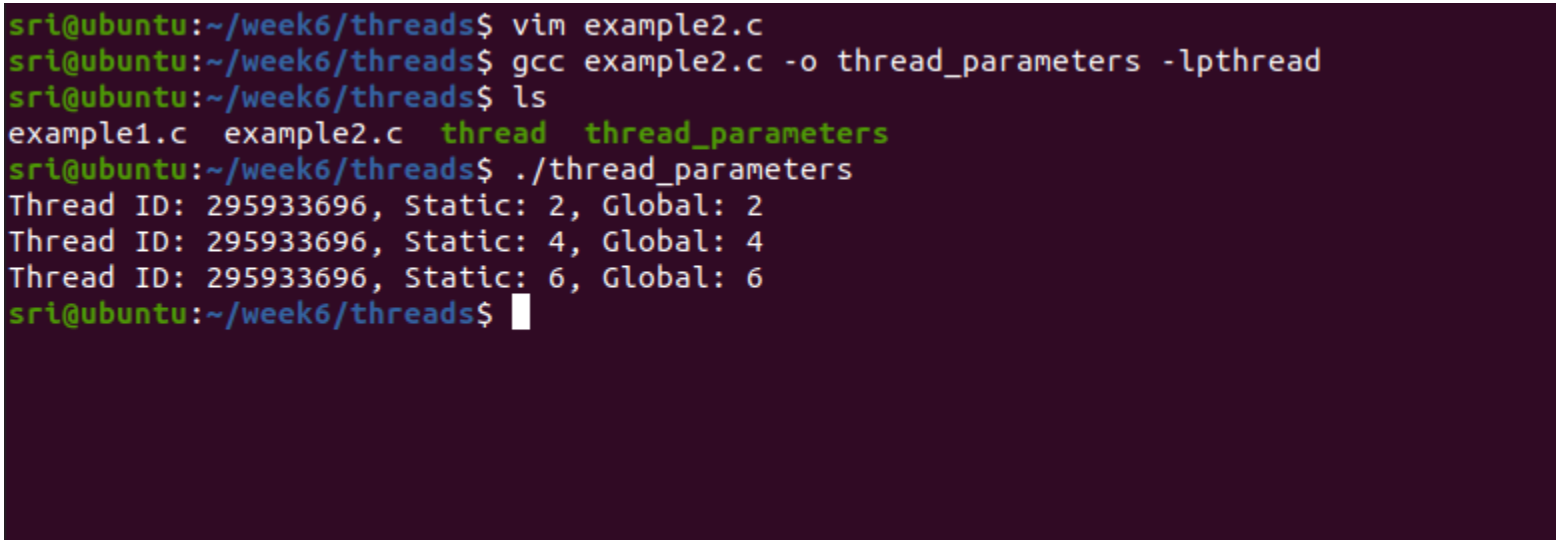
```
sri@ubuntu:~/week6/threads$ ls
example1.c
sri@ubuntu:~/week6/threads$ gcc example1.c -o thread -lpthread
sri@ubuntu:~/week6/threads$ ls
example1.c  thread
sri@ubuntu:~/week6/threads$ ./thread
Before Thread
Printing HELLO WORLD from Thread
After Thread
sri@ubuntu:~/week6/threads$
```

Example 2: Parameters

인하대학교 INHA UNIVERSITY

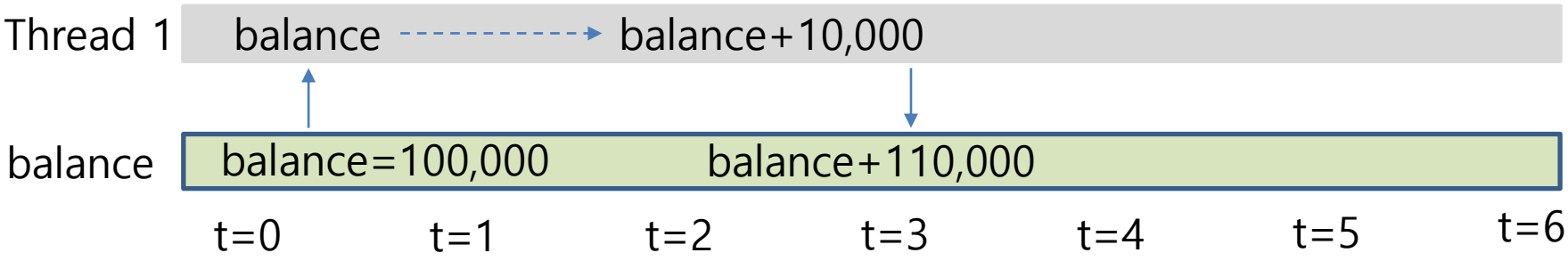
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

// Let us create a global variable to change it in threads
int g = 0;

// The function to be executed by all threads
void *myThreadFun(void *vargp)
{
        // Store the value argument passed to this thread
        int *myid = (int *)vargp;

        // Let us create a static variable to observe its changes
        static int s = 0;

        // Change static and global variables
        ++s; ++g;

        // Print the argument, static and global variables
        printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, ++g);
}

int main()
{
        int i;
        pthread_t tid;

        // Let us create three threads
        for (i = 0; i < 3; i++)
                pthread_create(&tid, NULL, myThreadFun, (void *)&tid);

        pthread_exit(NULL);
        return 0;
}
```

Execution


sri@ubuntu:~/week6/threads$ vim example2.c
sri@ubuntu:~/week6/threads$ gcc example2.c -o thread_parameters -lpthread
sri@ubuntu:~/week6/threads$ ls
example1.c example2.c thread thread_parameters
sri@ubuntu:~/week6/threads$ ./thread_parameters
Thread ID: 295933696, Static: 2, Global: 2
Thread ID: 295933696, Static: 4, Global: 4
Thread ID: 295933696, Static: 6, Global: 6
sri@ubuntu:~/week6/threads$

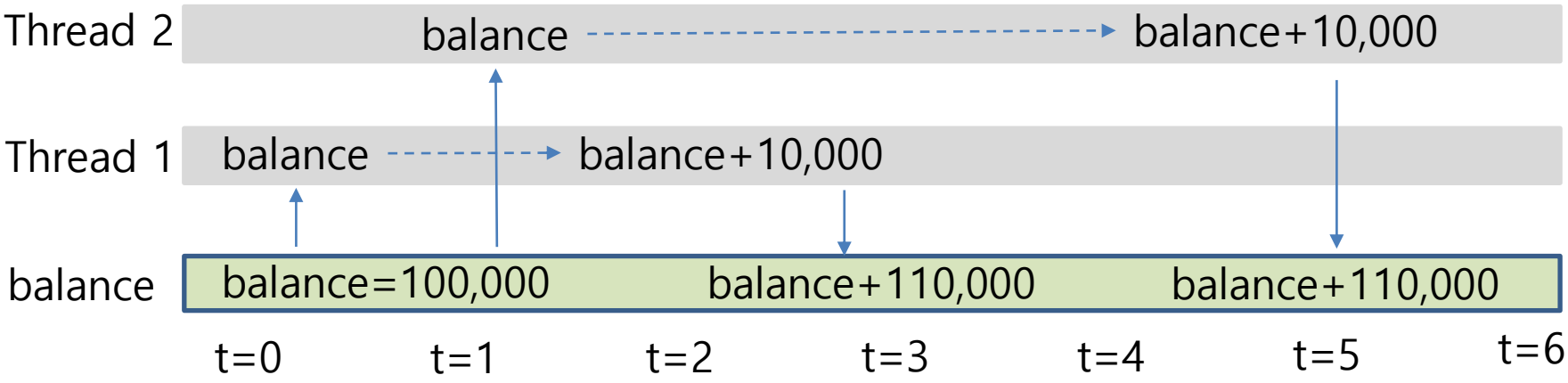
Discussion:

- How to safely access a shared variable in multi-threading
- Example: Imagine you have a bank account as global: **int balance**



Discussion: concurrency

- How to safely access a shared variable in multi-threading
- Example: Imagine you have a bank account as global: **int balance**



Discussion

- Most critical point of thread programming ➔ concurrency
- One should always be careful when accessing shared variables
- Accessing global variables should be avoided or …
- Accessing global variables should be controlled
- Solution: Locks (Mutex, Semaphores, etc.)

Practice

인하대학교 INHA UNIVERSITY

- Write a shell program to print all prime numbers between 1 to 50
  - Prime number: a number divisible only by 1 and itself
  - Example:
    - Prime numbers: 2, 3, 5, 7, 11 …
    - Non-prime numbers: 1, 4, 6, 8, 9, …

- Write a C program that calculates the sum of given values and returns the result:
  - You should give the values as parameter to the thread
  - Thread function should return the sum (value1 + value2)
  - The return value should be printed from main process

Questions?

인하대학교 INHA UNIVERSITY

Contact TA if you have questions
Name: Srinidhi
E-mail : ksrinidhi23@gmail.com
Intelligent Embedded System Lab. (H-813)