# Topic 1: Basic Concept

## Overview

### Guidance

Operating System Concepts 10th Edition.pdf chapter 1, chapter 20.1-20.3

Advanced Programming in the UNIX Environment, 3rd Edition.pdf 1.6 2.2 2.3

wiki:[History of Unix](https://en.wikipedia.org/wiki/History_of_Unix), [Linux distribution](https://en.wikipedia.org/wiki/Linux_distribution)

### Practice

1. Describe the following concepts

DMA, POSIX, Timer

1. How does the application program using the OS resources? What OS resources are there?

System resources are obtained by switching from user mode to kernel mode through system calls. The following resources can be used for processes, memory, files, storage, buffering, and I/O

1. What types of operating systems exist? What are their characteristics?

From different perspectives, OS can be divided into different types

* Categorized by user pattern

single-user and multi-user, <https://en.wikipedia.org/wiki/Operating_system>

* Categorized by task process pattern

single-tasking and multi-tasking:<https://en.wikipedia.org/wiki/Operating_system>

* Categorized by process mode

Batch Processing Operating System:<https://www.geeksforgeeks.org/batch-processing-operating-system/?ref=lbp>

Time Sharing Operating System:<https://www.geeksforgeeks.org/time-sharing-operating-system/?ref=lbp>

Real Time Operating System:<https://www.geeksforgeeks.org/real-time-operating-system-rtos/?ref=lbp>

* Categorized by application environment

Desktop Operating System:<https://prepinsta.com/operating-systems/desktop-systems/>

Mobile operating system:<https://en.wikipedia.org/wiki/Mobile_operating_system>

Server operating system:[https://cio-wiki.org/wiki/Server\_Operating\_System#google\_vignette](https://cio-wiki.org/wiki/Server_Operating_System" \l "google_vignette)

Embedded operating system:<https://en.wikipedia.org/wiki/Embedded_operating_system>

Network operating system:<https://en.wikipedia.org/wiki/Network_operating_system>

1. What is the relationship between unix and linux? What is the relationship between redhat and linux? What is the defference between linux destribution and linux kernel?

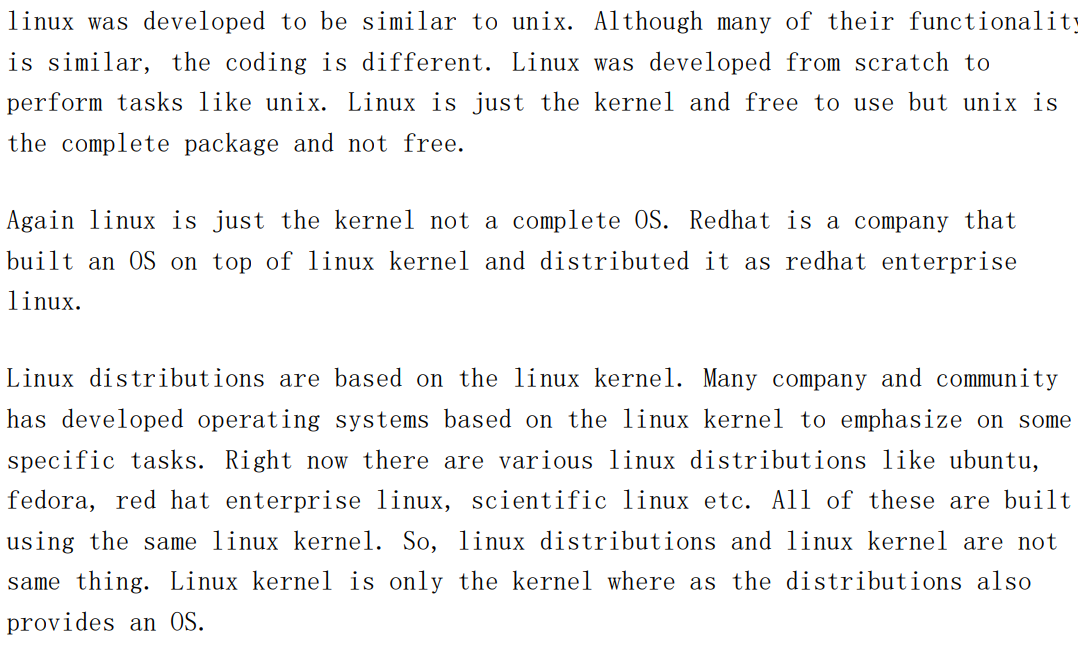
<https://upload.wikimedia.org/wikipedia/commons/7/77/Unix_history-simple.svg>

Linux is a Unix-like system,Just reference unix style design,No inheritance relationship with unix in source code

Linux distributions are based on the linux kernel, which is the linux we call usually.Linux consists of the linux kernel and supporting software.

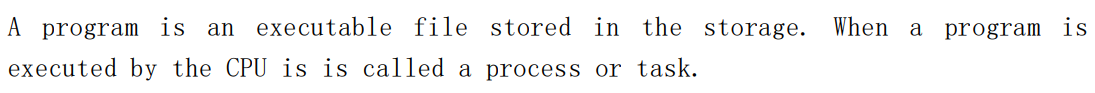
Linux kernel is a opensoure code, You can get it for free but unix is not.

From Dipto



1. What is the relationship between program and process?

From Saurav



1. Operating System Concepts 10th Edition.pdf chapter1 -> Practice Exercises 1.1
2. Operating System Concepts 10th Edition.pdf chapter1 -> Practice Exercises 1.3

## System Call

### Guidance

Operating System Concepts 10th Edition.pdf Chapter 2 System Calls

Advanced Programming in the UNIX Environment, 3rd Edition.pdf 1.11

ioctl dup

ulimit -a, What does these content mean? How to change it

### Practice

Write a program for each type of system call using the function in page 68.

If it can be done early by you, to explore more system calls and put the cases in your programs

## Process

### Guidance

Operating System Concepts 10th Edition.pdf 3.1-3.3

Operating Systems - Internals and Design Principles 7th.pdf 3.1-3.3

* What are similarity and difference between parent process and child process?
* ps/pstree
* Process state switch

### Practice

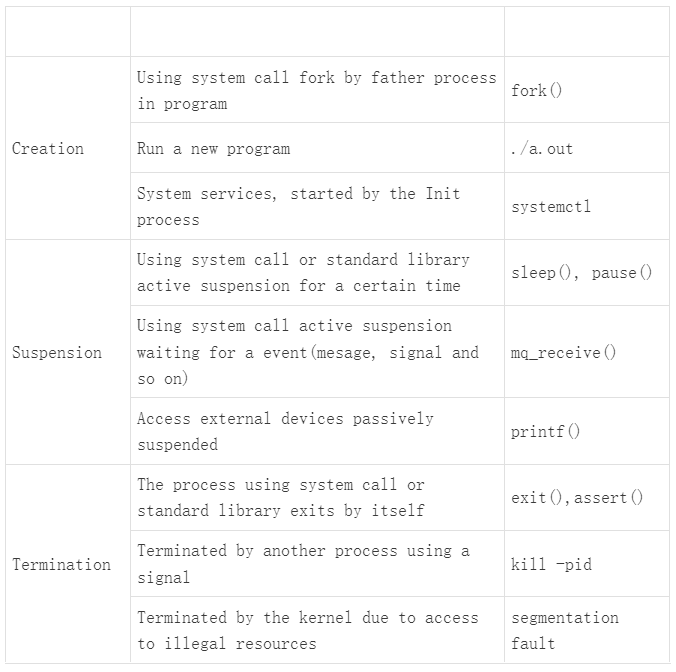
1. Run a program try to find it in pstree

Keep a process from ending, to find the process by command pstree

1. What is the meaning of each filed of command ps -elf?

There are 2 commands ps and top can be used to check the state and information of process and thread.

1. How many ways to create, suspense or terminate a process? Illustrate



1. Why do we need a PCB? What informations are stored in the PCB? What role does the PCB play when a process is created/scheduled/executed?

From Dipto

Why PCB is needed:PCB (process control block) is used to save the state of a process like it’sstatus, program counter, I/O information, registers used etc. On an OS thereare many processes. A CPU can not execute all of them concurrently. When aprocess has obtained CPU resource and in running state, it may be stopped(forI/O event or interrupt) abruptly without completing it’s task. While it iswaiting, some other process will get CPU resource. Since the first process isnot complete we need to continue executing it from the point it was halted. Forthis reason we need to save enough information so that it can be continuedexactly like before it was halted. This information is saved in PCB.

PCB saves information like: Process state: Like ready, running, wait, new and terminated. Program counter: Indicates the address of the next instruction to be executed CPU Register: Various registers used by the process like accumulator, stack pointer, index registers, general purpose registers etc. CPU-scheduling information: This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters. Memory-management information: This information may include suchitems as the value of the base and limit registers and the page tables, or thesegment tables. Accounting information: This information includes the amount of CPUand real time used, time limits, account numbers, job or process numbers,and so on.I/O status information: This information includes the list of I/O devicesallocated to the process, a list of open files etc

Roles of PCB:When a process is created, memory for it’s PCB is allocated. When it is in theready state(not in execution), it’s PCB is added to the tail of the readyqueue. When it gets CPU resources, it’s PCB is checked to see the informationof this process and execute accordingly.When it needs to be scheduled the current states are stored in the PCB and thenadded to the tail of wait queue. When it get back CPU resource again, the PCBis again checked to see it’s information before it got scheduled. As a result,it will be started from exactly where it left off.When the process is complete it will terminate and the memory of it’s PCB isdeallocated. But some information(like exit status) is saved temporarily untilthe parent of this process calls wait.

1. In what scenario switch between different states of a process？Illustrate

Refer to 3.2 on "Operating Systems - Internals and Design Principles 7th.pdf"

## Thread

### Guidance

Operating Systems - Internals and Design Principles 7th.pdf 4.1-4.2

Operating System Concepts 10th Edition.pdf 4.3-4.4

### Practice

1. For multiple threads in the same process, which resources are shared and which resources are unique?

* Resources exclusive by thread

stack frame and thread context,register,thread control block,Independent scheduling priority,error return code

note:The global variable can became thread exclusive after declared by \_\_thread.

parameters list and return value are thread exclusive without any declaration

* Resources shared by threads

global variable table(.text, .data, and so on), heap, dynamic-link library, files, Current working directory, as well as user ID and group ID

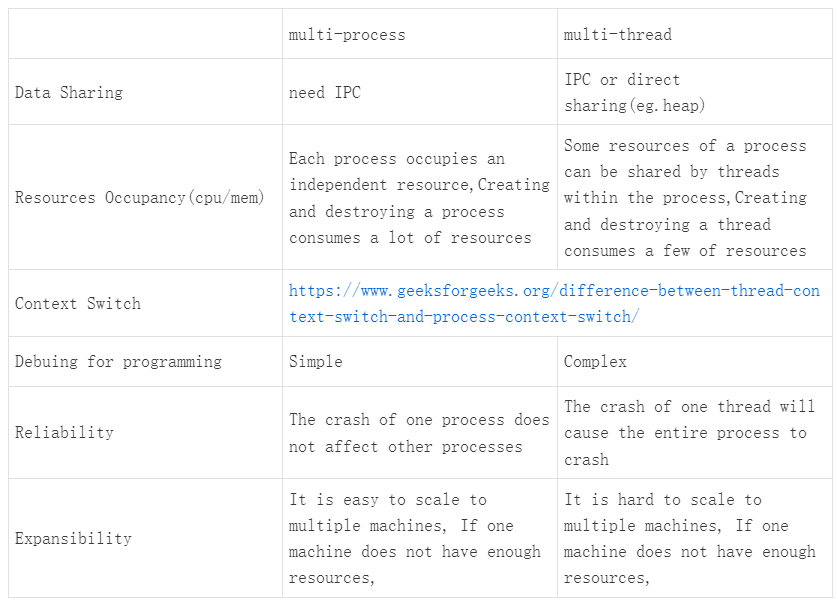
1. What is the thread type created in Linux using pthread\_create? ULT or KLT?

Both ULT and KLT, refet to p584 on "Operating Systems - Internals and Design Principles 7th.pdf"

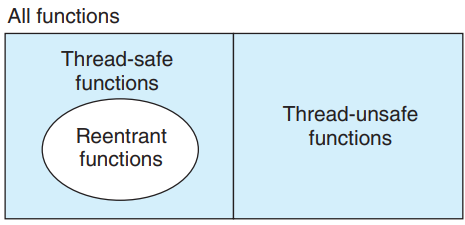
1. What are the advantages and disadvantags for multi-process(only one thread in each process) and multi-thread do programming?

process:Minimum unit of resource allocation

Thread:Minimum unit of cpu scheduling



1. Illustration



Reentrant functions and Thread-safe functions are different concept. Refer to p1059 on "Computer Systems A Programmers Perspective (3rd).pdf"

Reentrant functions: Refer to chapter 10.6/12.5 on Advanced Programming in the UNIX Environment, 3rd Edition.pdf

Thread-safe functions: When a function is called by different threads at the same time, each thread will get the correct result. This can only occur in multithreaded scenarios.

Reentrant functions: If a function is interrupted at any time during execution, the function can still be reused or the correct result can be obtained after the interrupt is returned. That is, it can occur in multithreaded or single-threaded scenarios.

Determine non-reentrant functions

* Functions that directly use or return global/static variables, including the errno variable
* malloc/free is called
* Standard I/O function is called，most of which use global data structures in a non-reentrant manner
* Other non-reentrant functions

A non-reentrant function can be locked to become thread-safe, but it's not reentrant. Reentrant functions are higher-level thread-safe functions, a concept similar to asynchronous - signal/interrupt safety

A function is reentrant if the data in it is in its own stack space.

1. Can multi thread share a routine function start\_routine? What should we pay attention to if we use it this way

Refer to man help, the rountine should be reentrant

1. How to make a thread never end?

while(1);

# Topic 2: IPC

## Basic Concept

### Guidance

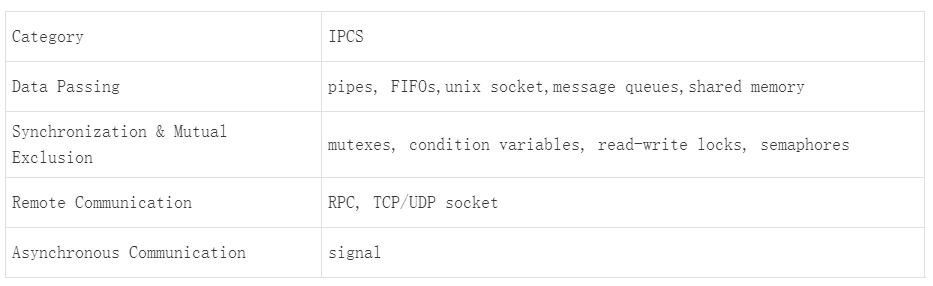
unix network programming Vol.2.pdf chapter 1-3

Advanced Programming in the UNIX Environment, 3rd Edition.pdf chapter 15

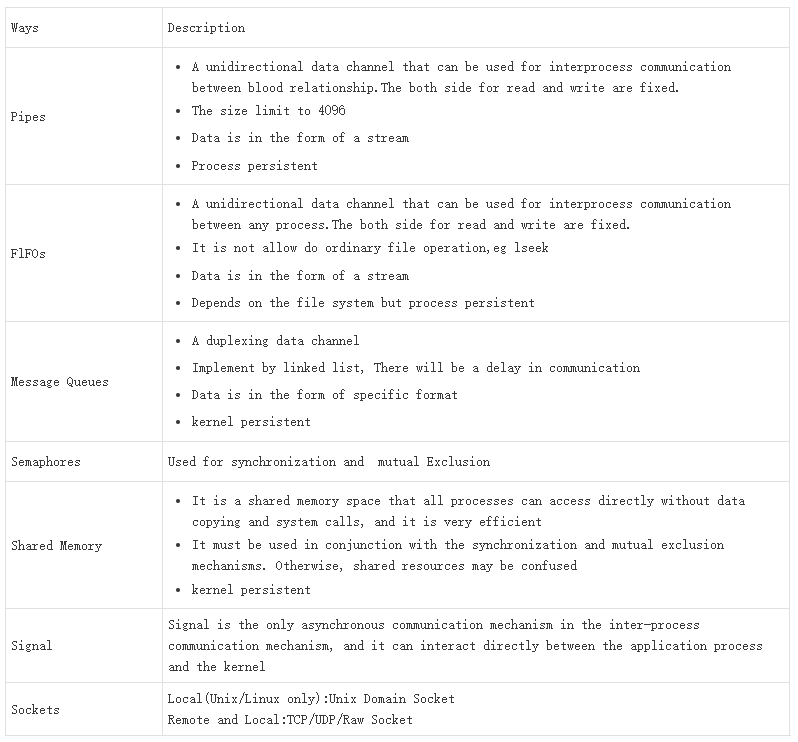
* The life cycle of the IPC object
* What is different between Posix and System V IPC?
* Can these ways be used for inter-thread communication

### Practice

1. According to your understanding, try to divide all IPCs into different categories, and the reason



1. To summarize, what are the characteristics of each IPC?



1. Which can be used for remote communication and which can be used for local communication. Can remote ways be used for local communication

Remote: Socket, It also can be used for local

Others: Local

## Data Passing

### Guidance

unix network programming Vol.2.pdf part 2

Advanced Programming in the UNIX Environment, 3rd Edition.pdf 17

pipes, FIFOs,unix socket,message queues,shared memory

### Practice

1. Use fork/exec simulation executive "ping 127.0.0.1 -c 3" and get the result

Refer to "fork-exec simulate executive ping.c"

1. Follow the client/server model, client sends a filename to server, response its content, implement by unix socket

Refer to "UDP based domain socket"

1. Follow the producer/consumer model, producer sends message MSG1/MSG2/MSG3 to consumer, print the messages when the consumer received it. Give a sample for the 2 kinds of message queue separately

## Synchronization & Mutual Exclusion

### Guidance

unix network programming Vol.2.pdf 7 10 11

mutexes, condition variables, read-write locks, semaphores

### Practice

Use semaphores(system v or posix) to achieve synchronization and mutual exclusion respectively

1. Write a program synchronizes by semaphore

Refer to "synchronization by semaphore.c"

1. Write a program that is mutually exclusive by semaphore

Refert to "mutual exclusion by semaphore.c"

1. Build several scenarios that generating deadlocks and implement them programmatically

## Remote Communication

### Guidance

Advanced Programming in the UNIX Environment, 3rd Edition.pdf 16

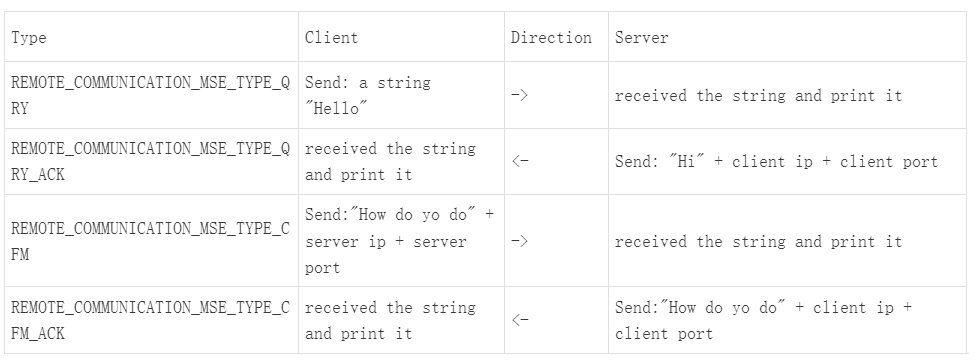
RPC, TCP/UDP socket. (These ways can also be used for local communication)

### Practice

1. Implement a 1-to-1 client/server program based on the following requirements.

The python code has already provided as the client which run in windows

Implement the server program code by C which run in linux



Refer to "UDP based socket"

1. Reconstruction question 2 on "Data Passing" make it can be usd for remote communication.

server in linux code by C

client in windows code by python

Refer to "Get file contents via UDP based socket"

## Asynchronous Communication

### Guidance

Advanced Programming in the UNIX Environment, 3rd Edition.pdf 10

signal

kill -l

### Practice

1. Summarize all signals

|  |  |  |  |
| --- | --- | --- | --- |
| Signal name | Signal number | Generation Mode | Generation scenario |
|  |  |  |  |

1. Explain the meaning of following conceptes

standard signal, real-time signal, reliable signal, unreliable signal

1. Modify the default behavior of ctrl + c when ctrl + c is pressed, the output is Hello World! ; Use the kill command to achieve the same effect
2. Ignore the signal generated by ctrl + c and run while(1); How to terminate a process at this point; Provide another example of ignoring other signals
3. Which system calls can standard signals? For what scenarios? What signal is being generated? List at least 5
4. Implement sleep() and a timer using alarm() and pause() simulations. And wake up the process in the sleep state in advance in the timer

Refer to “sleep and timer.c”

# Topic 3: VxWorks Task

## Task

### Guidance

Vxworks\_programmers\_guide5.5.pdf 2.2

### Practice

Experiment

1. Create and run a task when switch boot
   1. Create and run a task named DEMO1 with stack size 1024 bytes and priority 128 in function os\_demo\_init. Print The task ID and name here
   2. The start routine of DEMO1 is os\_demo\_main\_process. There is a main loop in the function. It will be active interval 10s.
   3. There is a global variable named g\_int\_data. Which will be increase 1 and print the task ID and name every time when the main loop is actived.
2. Create and run a task when execute command
   1. Create and run a task named DEMO2 with stack size 1024 bytes and priority 128 in function os\_demo\_cmd\_impl\_test1 of file "os-demo-cmd-impl.c", when command "os-demo test1" be executed. Print The task ID and name here
   2. The start routine of DEMO2 is os\_demo\_cmd\_impl\_test1\_process. There is a loop in the function. It will be active interval 30s.
   3. print the task ID, name and the global variable g\_int\_data every time when the loop is actived.
3. Destory specify task
   1. Reconstruction function second\_os\_demo\_config\_test2 to accept a number
   2. Delete a specify task id which is prefixed with DEMO

Observation and Thinking

1. How many task names are there output? List out them
2. Is it possible for DEMO1 and DEMO2 to access both the global variable g\_int\_data? Why?
3. What is the meaning of in following fields that display on "show task"?

NAME ENTRY TID PRI CPU invoked

1. Can the command "os-demo test1" be executed multi-times? using command "show task | include DEMO" to check
2. When deleting a task, what happens if the task does not exist? How to prevent this phenomenon from happening

Refer to “Task creation and deletion”

## Inter Task Communication

### Guidance

Vxworks\_programmers\_guide5.5.pdf 2.3, 2.4

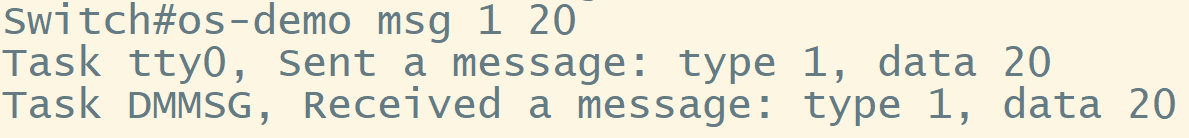
### Practice

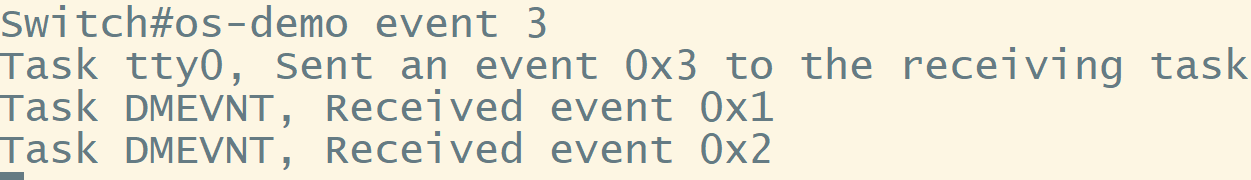
If not specified, the default values for the tasks are as follows:

stack size is 1024 bytes, priority 128, No parameter

Experiment for message and event in different task

1. Messaage and Event receiving
   1. Create a message queue Which len is 20,as MSG\_Q\_FIFO mode
   2. Create and run a task named DMMSG,Which start routine is os\_demo\_msg\_recv and used for receiving message.
   3. Create and run a task named DMEVNT,Which start routine is os\_demo\_event\_recv and used for receiving event.
2. Messaage and Event sending
   1. Send a message by command line "os-demo msg "
   2. Send a event by command line "os-demo event "
3. Process demonstration





Refer to “Receiving message and event in a same task”

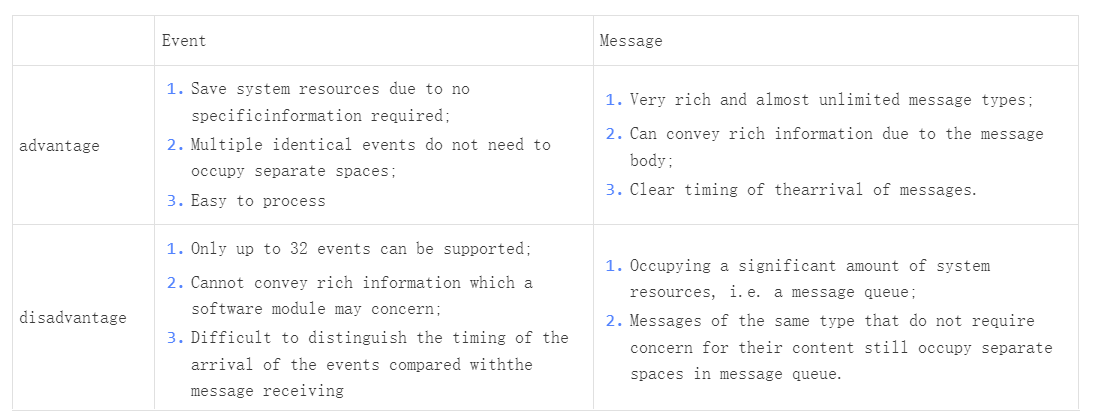
Experiment for message and event in a same task

1. Messaage and Event receiving
   1. Create a message queue Which len is 20,as MSG\_Q\_FIFO mode
   2. Create and run a task named DMMSGEVNT, Which start routine is os\_demo\_msg\_event\_recv and used for receiving message and event.
2. Messaage and Event sending
   1. Send a message by command line "os-demo msg <type> <value>"
   2. Send a event by command line "os-demo event <event>"

Hint: A task can only have one block point

Observation and Thinking

How many solutions can you though? What are the advantage, disadvantage and difference between them?



Refer to “Receiving message and event in a same task”

Experiment for multiple socket in different task

Refer to "socket.c" and file-write-read for VX switch.c, Reconstruct the 2 experiments in Remote Communication of Topic 2

1. Create and run a task named SCKT,Which start routine is os\_demo\_socket\_file\_process, Which function refer to question 1
2. Create and run a task named SCKF,Which start routine is os\_demo\_socket\_file\_process, Which function refer to question 2
3. Use the client on windows to communicate with the switch

Refer to "Receiving multiple socket in different task"

Experiment for multiple socket in a same task

Based on system call select() to reconstruct "Experiment for multiple socket in different task".

There is a task which name is SKTF and start routine is os\_demo\_socket\_text\_file. It's waiting for reponse from file request and text request

Experiment for mesage, event and multiple socket in a same task

Based on socket\_register() to reconstruct "Experiment for multiple socket in a same task" and "Experiment for message and event in a same task"

There is a task which name is SKEM and start routine is os\_demo\_socket\_message\_event\_process, stack size is 1024 \* 8. It's waiting for reponse from message, event and multiple socket

## Timer

### Guidance

#include <libsys/timer.h>

bdip-task.c

### Practice

Reconstruct the experiment "Experiment for message and event in a same task", Use the timer instead of command line to periodically send event and message to task the DMMSGEVNT.

1. Create and start a timer named timer\_event, which is loop and callback type. Sent a event to the task DMMSGEVNT every 15 seconds
2. Create and start a timer named timer\_msg, which is loop and message type. Sent a message to the task DMMSGEVNT every 15 seconds
3. Print the infomations of sender and receiver.

eg.

Task , Sent a mesasge: type <type>, data <data>

Task , Received a message: type <type>, data <type>

Observation and Thinking

1. What is the difference between the 2 timers?
2. Can you infer how the timer works?
3. What do you need to pay attention to when you do something in timer? (Resource management perspective)

Experiment for timer application

1. There is a linked list,which stored many ip cache nodes,Refer to Sample 1 for the struct
2. There is a task which name is DMCR, start routine is os\_demo\_ip\_cache\_create\_process used for creat a ip cache node and add it to the linked list.Every random time(refer to sample 2). IP address is also generated by random(refer to sample 3).
3. There is a task Which name is DMMC, start routine is os\_demo\_ip\_cache\_get\_mac\_process used for provide the mac address for appointed ip address when DMCR is creaing the ip cache node
4. There is a task Which name is DMAG, Start routine is os\_demo\_ip\_cache\_age\_check\_process used for increase 1 to field age for each ip cache node periodic(implement by timer) and delete the node when the age reach to a 100.



## Task Scheduling

\* Experiment for the preemptive priority scheduling

Get the corresponding code from Trainer and run it

There are 3 tasks:

a)Counting task with priority 128. It has two parts of code in its main loop: one is a non blocking code block implemented as a dual loop, in which the value of a global variable “test\_count” is incremented; and the other is just sleeping by calling taskDelay().

b)A checking task with the same priority of 128. It periodically checks the value of test\_count and prints it out.

c)A checking task with the a higher priority of 126. It also periodically checks the value of test\_count and prints it out. Read the source code for the detail.

Observation and Thinking

What do you observe? Try to explain

Task DNCK always prints out 640000 while task DHCK prints out a value that varies from 0 to 640000 (the probability of printing out 0 or 640000 is very low for task DHCK).

Under the preemptive priority scheduling, a task with a higher priority entering the ready state always preempts the CPU from the current task running. However, a task with the same priority cannot preempt CPU from the running one.

\* Experiment for CPU Hog

Get the corresponding code from Trainer and run it

If a task is overloaded after receiving a message. Although there may not be such a simple dual loop logic in the real system, the situation is similar when dealing with large capacity tables.

Observation and Thinking

1. What do you observe? Try to explain

Unable to interact with the switch through console and telnet。Because of the CPU hog caused by the task DMRV, the other tasks have no chance to run.

1. How to optimize this code to eliminate CPU hog

To yield the CPU for a moment each time every some seconds using tickGet().

At least the absolute CPU hog is eliminated, and we can access the switch by console and telnet session although the interaction in the sessions is delayed for response. Because the task DMRV will be in the delay state (i.e. yields the CPU) for a while every time it runs for 1.5 seconds, the relevant tasks now have chance to run.

# Topic 4: OS in Switch

### Alignment

To change the program "Receiving multiple socket in different task**",**Parse received data without using struct pointers, print out the type, length and data

### MBLK Leak

Experiment

To change to program "Receiving message, event and multiple socket in a same task" don't to take the data away from socket when task received the corresponded message from message queue. Add a print when received a message is received.

Sent a text and file data to switch. The corresponding message is printed. Do not operate the switch. Wait for one hour

Observation and Thinking

What did you observe? What's the meaning of the output?