## Network Computing & Programming

# Lecture 2

# Network Programming Fundamentals

**2019-2020-1**

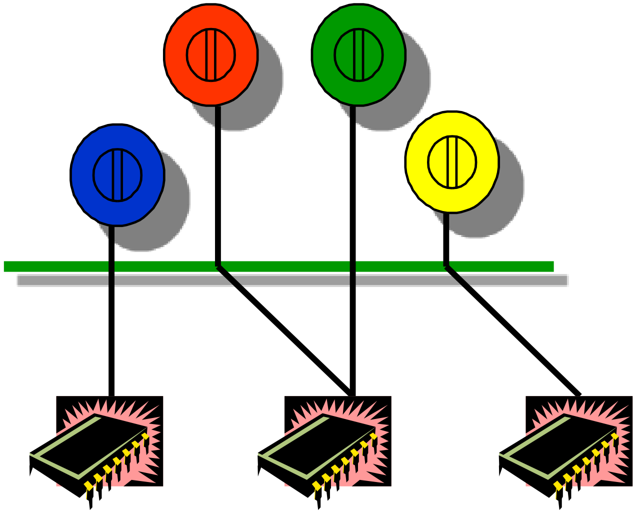
**Slides are modified from Xiang Zhang**

# Content

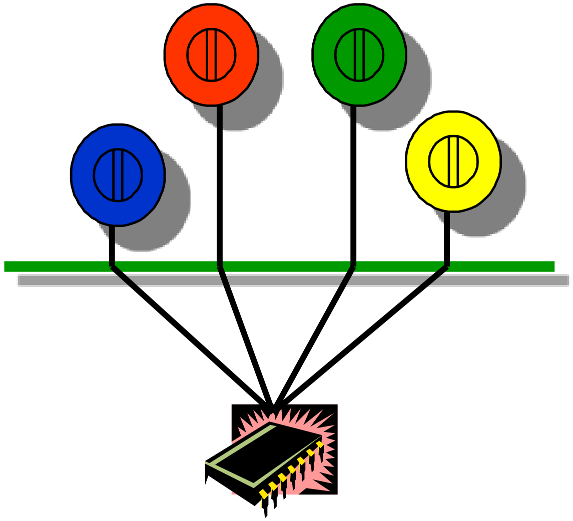
- ❑ Concurrence & Parallelism
- ❑ Byte Ordering (Endianness)
- ❑ ILP/LP/LLP Definition
- ❑ Memory Alignment
- ❑ Protocol Header Data Structure
- ❑ Development & Debug Tools

# Concurrence & Parallelism

- ❑ **Parallelism** (并行)
- ❑ **Concurrency** (并发，看起来同时发生)



并行模型



并发模型

# Concurrency in Networks & Hosts

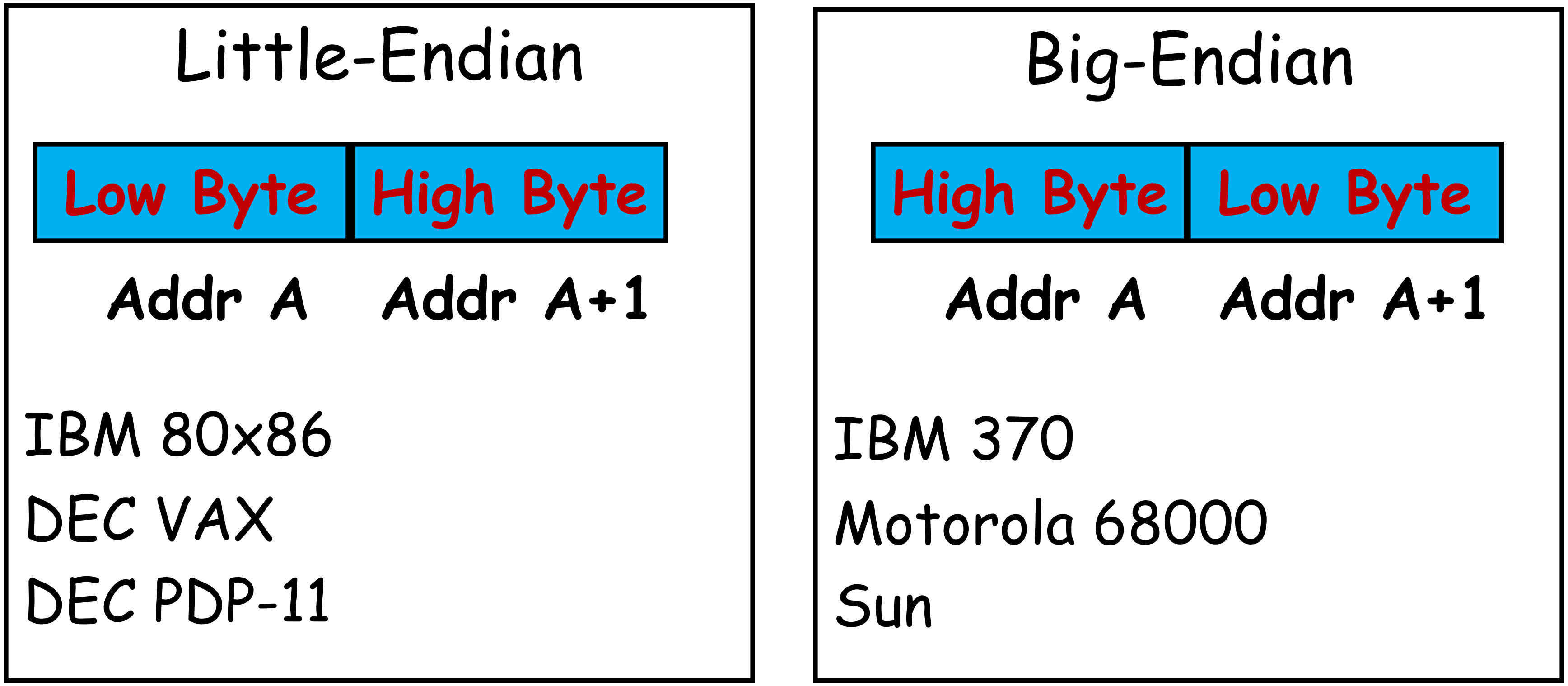
- ❑ Peer-to-peer processes between two hosts work concurrently.
- ❑ All clients in the same host work concurrently.
- ❑ All requests introduced into a server can be processed concurrently.
- ❑ ……

# Concurrence & Parallelism

- Multiprogramming
  - Tasks multiplex their executions on a single processor
- Multiprocessing
  - Tasks multiplex their executions on a multiprocessor system where there is access to shared memory
- Distributed Processing
  - Tasks multiplex their executions on several processors which do not share memory

# Byte Ordering

- Different computer architectures use different byte ordering to represent multi-byte values.



Little-Endian

| Low Byte | High Byte |
|---|---|
| Addr A | Addr A+1 |

IBM 80x86
DEC VAX
DEC PDP-11

Big-Endian

| High Byte | Low Byte |
|---|---|
| Addr A | Addr A+1 |

IBM 370
Motorola 68000
Sun

# Byte Ordering

- Suppose a Big Endian machine sends a 16 bit integer with the value 2:

0000000000000010

- A Little Endian machine will think it got the number 512:

0000001000000000

# Network Byte Order

- How do lower level layers communicate if they all represent values differently ? (data length fields in headers)
- A fixed byte order is used (called *network byte order*) for all control data.
  - TCP/IP : big-endian order

# Byte Order Dection

```
#include "stdio.h"
int main(int argc, char **argv)
{
      union{
                   short   s;
                   char    c[sizeof(short)];
             } un;
      un.s = 0x0102;
      if (sizeof(short) == 2) {
             if (un.c[0] == 1 && un.c[1] == 2)
                   printf("big-endian\n");
             else if (un.c[0] == 2 && un.c[1] == 1)
                   printf("little-endian\n");
             else  printf("unknown\n");
             }
      else
             printf("sizeof(short) = %d\n",
  sizeof(short));
             exit(0);
}
```

# Byte Order Conversion

- Convert multi-byte integer types from host byte order to network byte order

| | | |
|---|---|---|
| ① | htons() | host to network short |
| ② | htonl() | host to network long |
| ③ | ntohs() | network to host short |
| ④ | ntohl() | network to host long |

- <arpa/inet.h>

# Quiz

- Please write out 192.168.128.64 by Big-Endian & Little-Endian & Network Byte Order rules separately.

- If we want to send out a message with a destination IP address 222.111.0.1, which is stored by Little-Endian rule, what shall we do?

# Quiz

- Two struct definitions
  - struct { char a; char b; };
  - struct { int a; int b; };
- Considering Big-Endian and Little-Endian rules separately.
  - **If to read memory from the low address to the high address by the size of type, is a available firstly or secondly?**
  - **If to transmit bytes into network from the low address to the high address one by one, is a transmitted firstly or secondly?**

# Data Type Models: ILP/LP/LLP

- ILP
  - integers (I), long integers (L), and pointers (P)
  - Linux/Unix: LP64; Windows: LLP64

| | LP32 | ILP32 | LP64 | LLP64 | ILP64 |
|---|---|---|---|---|---|
| char | 8 | 8 | 8 | 8 | 8 |
| short | 16 | 16 | 16 | 16 | 16 |
| int | 16 | 32 | 32 | 32 | 64 |
| long | 32 | 32 | 64 | 32 | 64 |
| long long | 64 | 64 | 64 | 64 | 64 |
| pointer | 32 | 32 | 64 | 64 | 64 |

# Memory Alignment

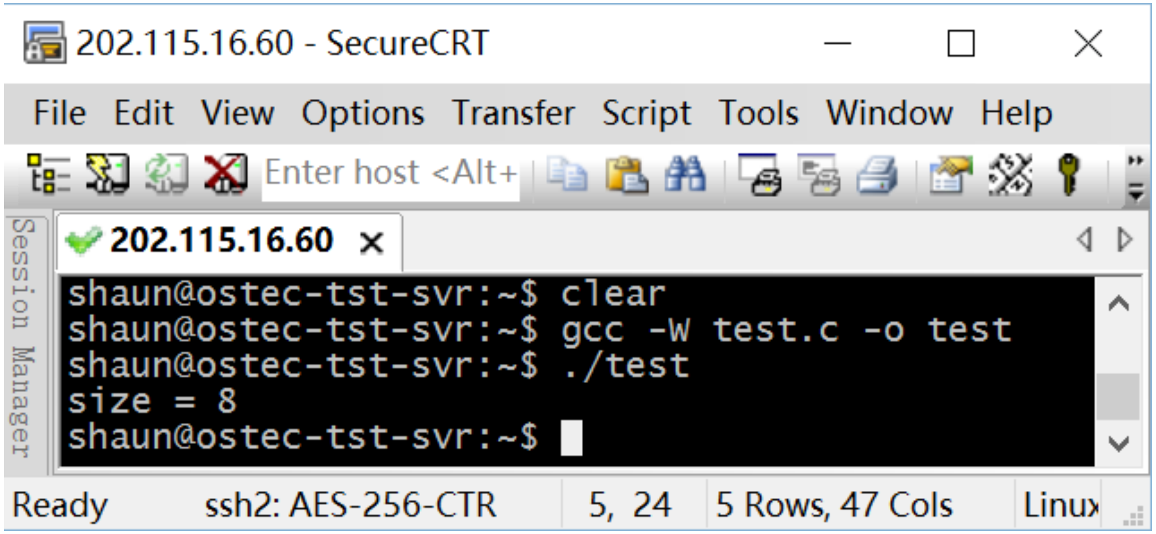
- Take a look at the codes

```
#include <stdio.h>

struct Test
{
        char x1;
        int x2;
};

int main(int argc, char *argv[])
{
        int size = sizeof(struct Test);
        printf("size = %d\n", size);

        return 0;
}
```



```
shaun@ostec-tst-svr:~$ clear
shaun@ostec-tst-svr:~$ gcc -W test.c -o test
shaun@ostec-tst-svr:~$ ./test
size = 8
shaun@ostec-tst-svr:~$ 
```

# Memory Alignment

- **How to read a 4-byte value stored at an address that is not a multiple of 4 bytes?**
  - i386 permits this kind of access. But **NOT ALL** architectures permit it
    - Can raise exceptions
- **Portability of data structures**
  - Compiler rearranges structure fields to be aligned according to platform-specific conventions
  - Automatically add padding to make things aligned
    - May no longer match the intended format

# Memory Alignment

- Another example, consider the following structure on a 32-bit machine

```
struct animal_struct
{
       char dog;                  /* 1 byte    */
       unsigned long cat;         /* 4 bytes   */
       unsigned short pig;        /* 2 bytes   */
       char fox;                  /* 1 byte    */
};
```

# Memory Alignment

- Structure not laid out like that in memory
  - Natural alignment of structure's members is inefficient
- Instead, complier creates padding

```
struct animal_struct {
        char dog;                 /* 1 byte    */
        u8 __pad0[3];             /* 3 bytes   */
        unsigned long cat;        /* 4 bytes   */
        unsigned short pig;       /* 2 bytes   */
        char fox;                 /* 1 byte    */
        u8 __pad1;                /* 1 byte    */
};
```

# Memory Alignment

- You can often rearrange the order of members in a structure to obviate the need for padding

```
struct animal_struct {
       unsigned long cat;  /* 4 bytes   */
       unsigned short pig; /* 2 bytes   */
       char dog;           /* 1 byte    */
       char fox;           /* 1 byte    */
};
```

# Memory Alignment

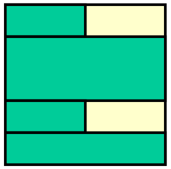
- Another option is to tell the compiler to pack the data structure with **NO PADDING** added
- Example: `<linux/edd.h>`

```
struct {
  u16 id;
  u64 lun;
  u16 reserved1;
  u32 reserved2;
} __attribute__ ((packed)) scsi;
```

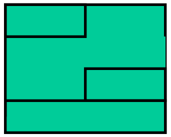
- Quiz: What is the size of scsi on systems with ILP32 or LP64?

# Memory Alignment

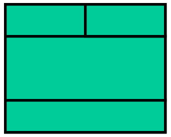
- No compiler optimizations



- __attribute__ ((packed))



- Some compiler optimizations



# Memory Alignment

- **#pragma pack**
  - Each system platform has its own Memory Alignment Coefficient.
  - Preprocessor directives can help to alter this value.
    - **#pragma pack(n), n=1,2,4,8,16 bytes.**
- Rule:
  - We need to carefully design the data structure to make full use of the memory and try not padding when doing network programming.

# Quiz

- What are the exact size on x32 Linux system with default pack number 4.
  - struct X1 { char a; int b; char c; };
  - struct X2 { int a; char b; };
  - struct X3 { char a; short b; };
  - struct X4 { char a; short b; char c; };
  - struct X5 { char a; long b; };
  - struct X6 { char a; long b; char c; };
  - struct X7 { char a; long long b; };
  - struct X8 { char a; long long b; char c; };
  - struct X9 { char a; int b; short c; };
  - struct X10 { char a; short b; char c; int d; };

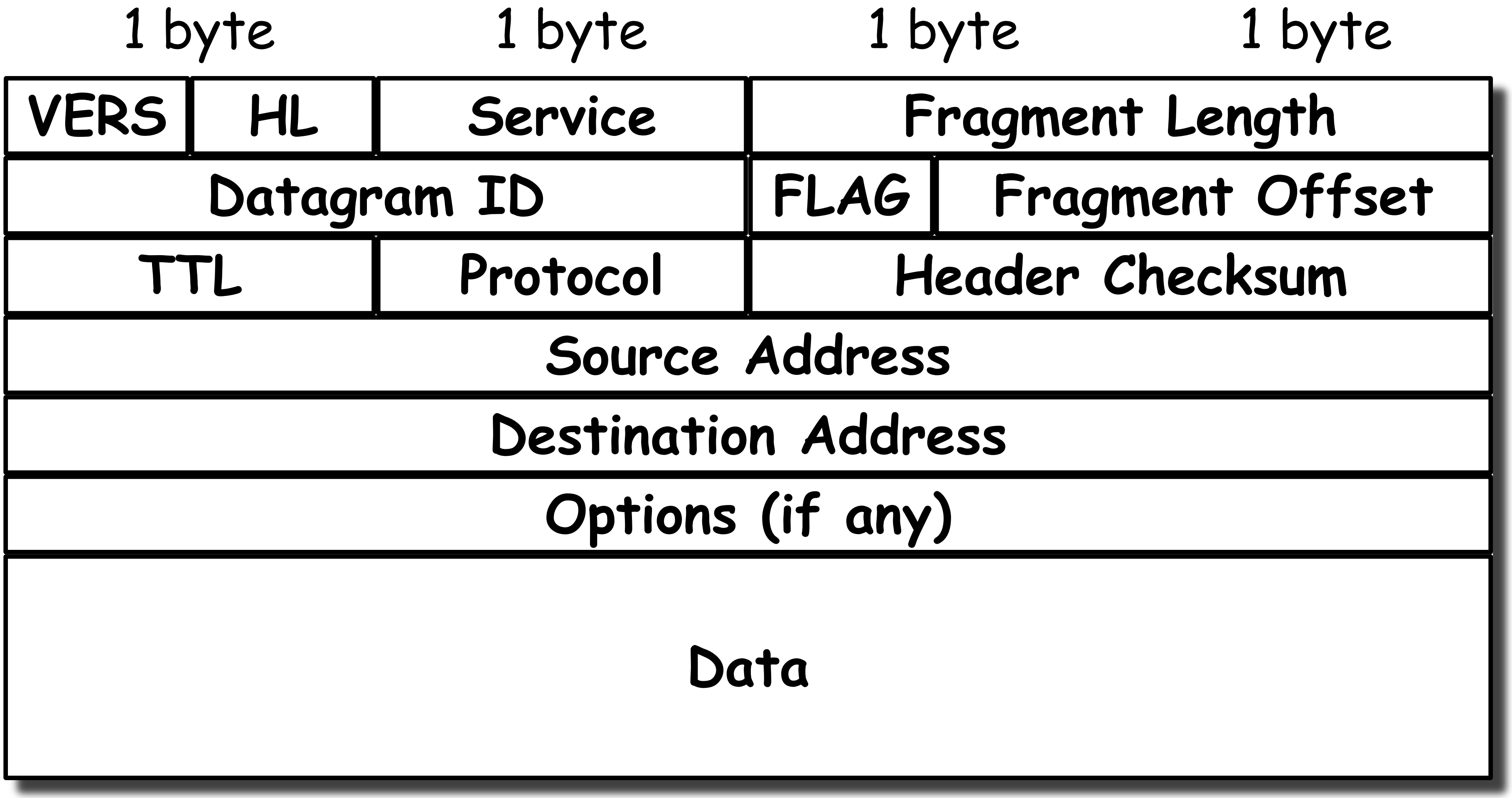
# Protocol Header Data Structure-Ethernet

| Preamble | Destination Address | Source Address | Len | DATA | CRC |
|---|---|---|---|---|---|
| 8 Bytes | 6 | 6 | 2 | 0-1500 | 4 |

```
<linux/if_ether.h>

struct ethhdr {
        unsigned char   h_dest[ETH_ALEN];
            /* destination eth addr */
        unsigned char   h_source[ETH_ALEN];
           /* source ether addr    */
        __be16          h_proto;
           /* packet type ID field */
} __attribute__((packed));
```

# Protocol Header Data Structure-IP



1 byte 1 byte 1 byte 1 byte

VERS HL Service Fragment Length

Datagram ID FLAG Fragment Offset

TTL Protocol Header Checksum

Source Address

Destination Address

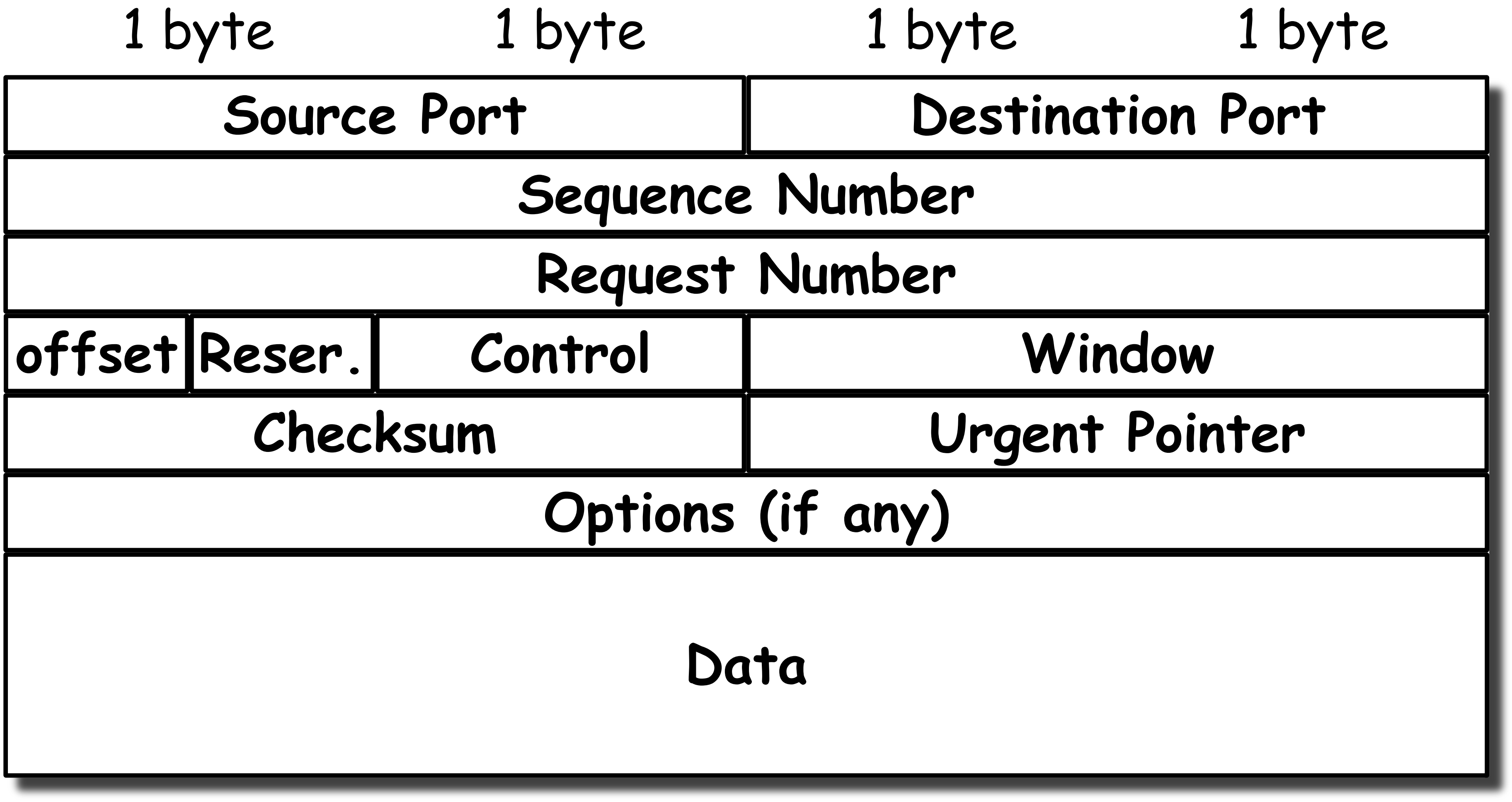
Options (if any)

Data

# Protocol Header Data Structure-IP

```
                                 <linux/ip.h>
struct iphdr {
  #if defined(__LITTLE_ENDIAN_BITFIELD)
        __u8 ihl:4, version:4;
  #elif defined (__BIG_ENDIAN_BITFIELD)
       __u8 version:4, ihl:4;
  #else
  #error "Please fix <asm/byteorder.h>"
  #endif
        __u8   tos;                 /*服务类型*/
        __be16 tot_len;             /*总长度*/
        __be16  id;                 /*标识*/
        __be16 frag_off;            /*片偏移*/
        __u8   ttl;                 /*生存时间*/
        __u8   protocol;            /*协议类型*/
        __u16  check;               /*头部校验和*/
        __be32 saddr;               /*源IP地址*/
        __be32 daddr;               /*目的IP地址*/
  };
```

# Protocol Header Data Structure-TCP



<table>
<tr><th colspan="2">1 byte</th><th colspan="2">1 byte</th><th colspan="2">1 byte</th><th colspan="2">1 byte</th></tr>
<tr><td colspan="4">Source Port</td><td colspan="4">Destination Port</td></tr>
<tr><td colspan="8">Sequence Number</td></tr>
<tr><td colspan="8">Request Number</td></tr>
<tr><td>offset</td><td>Reser.</td><td colspan="2">Control</td><td colspan="4">Window</td></tr>
<tr><td colspan="4">Checksum</td><td colspan="4">Urgent Pointer</td></tr>
<tr><td colspan="8">Options (if any)</td></tr>
<tr><td colspan="8">Data</td></tr>
</table>

# Protocol Header Data Structure-TCP

<linux/tcp.h>

```
struct tcphdr {
    __be16 source;
    __be16 dest;
    __be32 seq;
    __be32 ack_seq;
#if
defined(__LITTLE_ENDIAN_BITFIELD)
    __u16   res1:4,
            doff:4,
            fin:1,
            syn:1,
            rst:1,
            psh:1,
            ack:1,
            urg:1,
            ece:1,
            cwr:1;
#elif
defined(__BIG_ENDIAN_BITFIELD)
    __u16   doff:4,
            res1:4,
            cwr:1,
            ece:1,
            urg:1,
            ack:1,
            psh:1,
            rst:1,
            syn:1,
            fin:1;
#else
#error "Adjust your
<asm/byteorder.h> defines"
#endif
    __be16 window;
    __be16 check;
    __be16 urg_ptr;
};
```

# Protocol Header Data Structure-UDP

| 1 byte 1 byte | 1 byte 1 byte |
|---|---|
| Source Port | Destination Port |
| Length | Checksum |
| Data | |

```
<linux/udp.h>
```

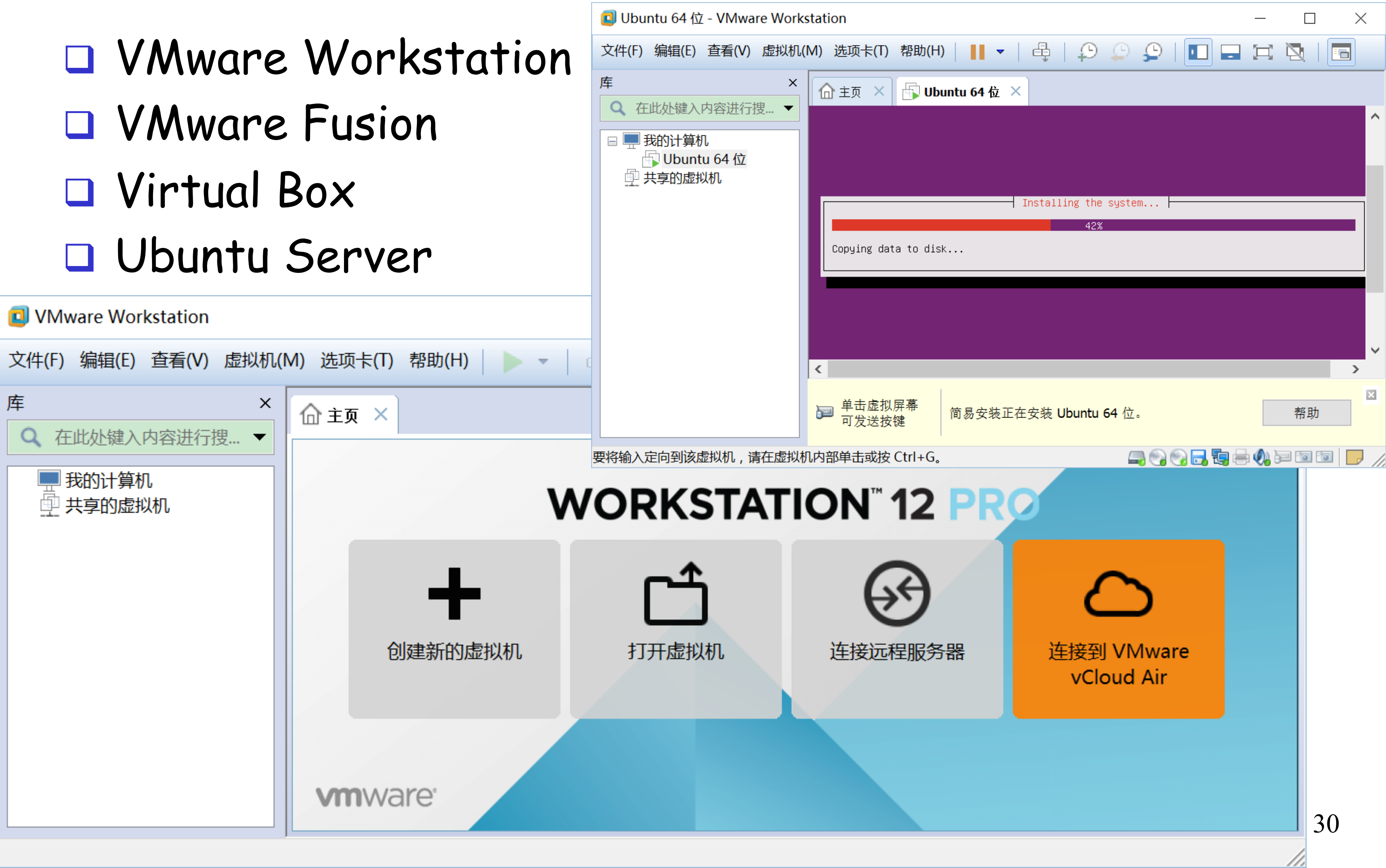
```
    struct udphdr {
            __be16  source;
            __be16  dest;
            __be16  len;
            __sum16 check;
    };
```

# Development & Debug Tools

- VMWare / Virtual box + Ubuntu Server
- GCC + GDB / Clang + LLDB
- netstat， ping， tcpdump

# Development & Debug Tools

- ❑ VMware Workstation
- ❑ VMware Fusion
- ❑ Virtual Box
- ❑ Ubuntu Server



Installing the system...

42%

Copying data to disk...

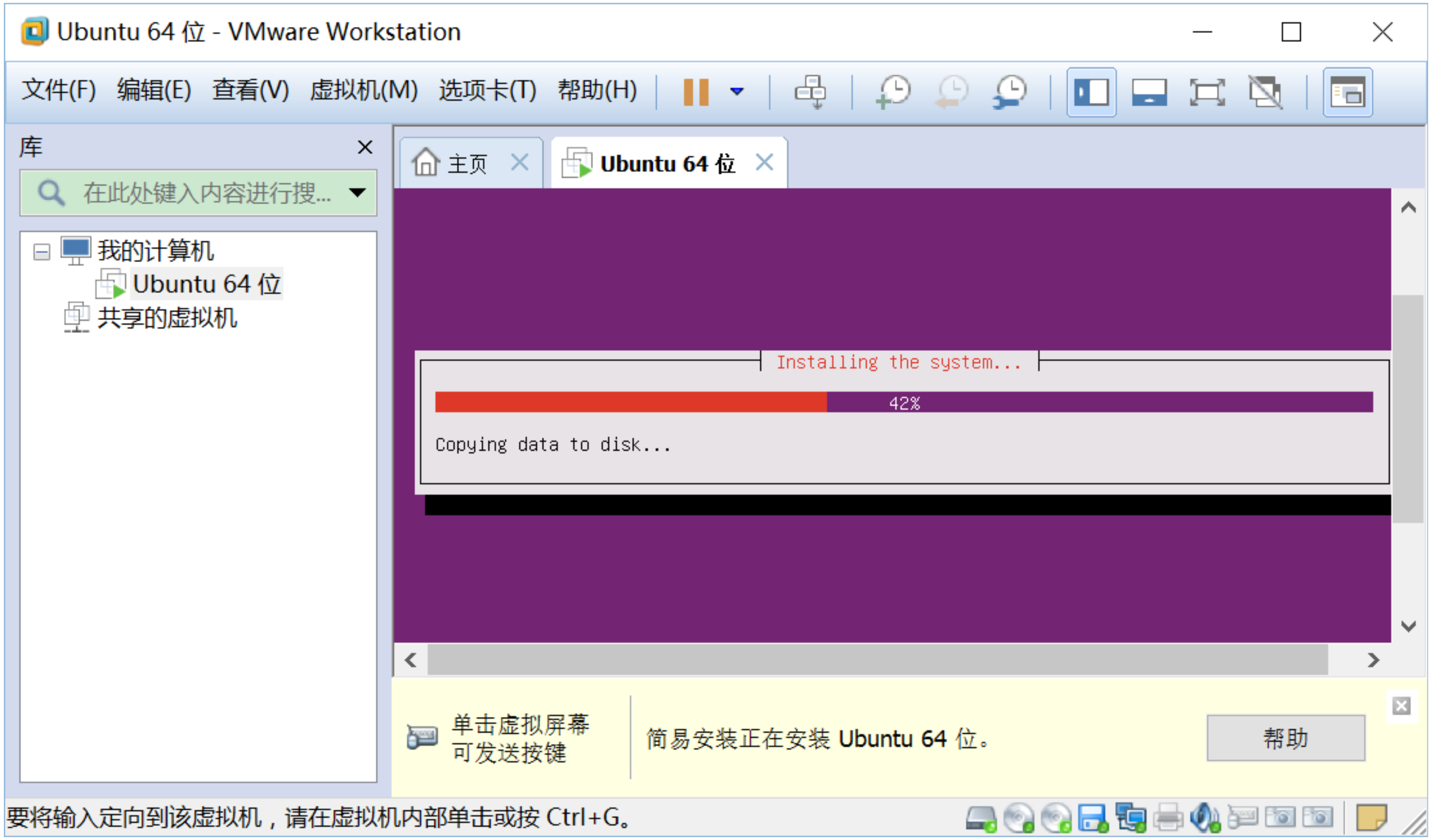
单击虚拟屏幕
可发送按键

简易安装正在安装 Ubuntu 64 位。

要将输入定向到该虚拟机，请在虚拟机内部单击或按 Ctrl+G。

# Development & Debug Tools

- ❑ Ubuntu x64



Ubuntu 64 位 - VMware Workstation

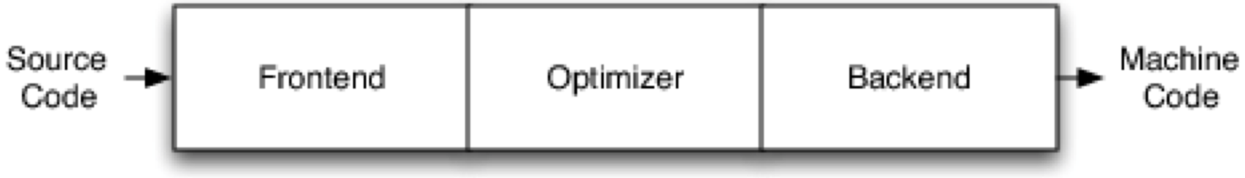
```
Installing the system...
42%
Copying data to disk...
```

简易安装正在安装 Ubuntu 64 位。

要将输入定向到该虚拟机，请在虚拟机内部单击或按 Ctrl+G。

# Development & Debug Tools

- **GCC (GNU Compiler Collection)**
  - Includes front ends for C, C++, Objective-C, Fortran, Ada, and Go, as well as libraries for these languages (libstdc++,...).
  - http://gcc.gnu.org
- **LLVM (Low Level Virtual Machine)**
  - Written in C++ and is designed for compile-time, link-time, run-time, and "idle-time" optimization of programs written in arbitrary programming languages
  - **Clang: an "LLVM native" C/C++/Objective-**



Source Code

Frontend

Optimizer

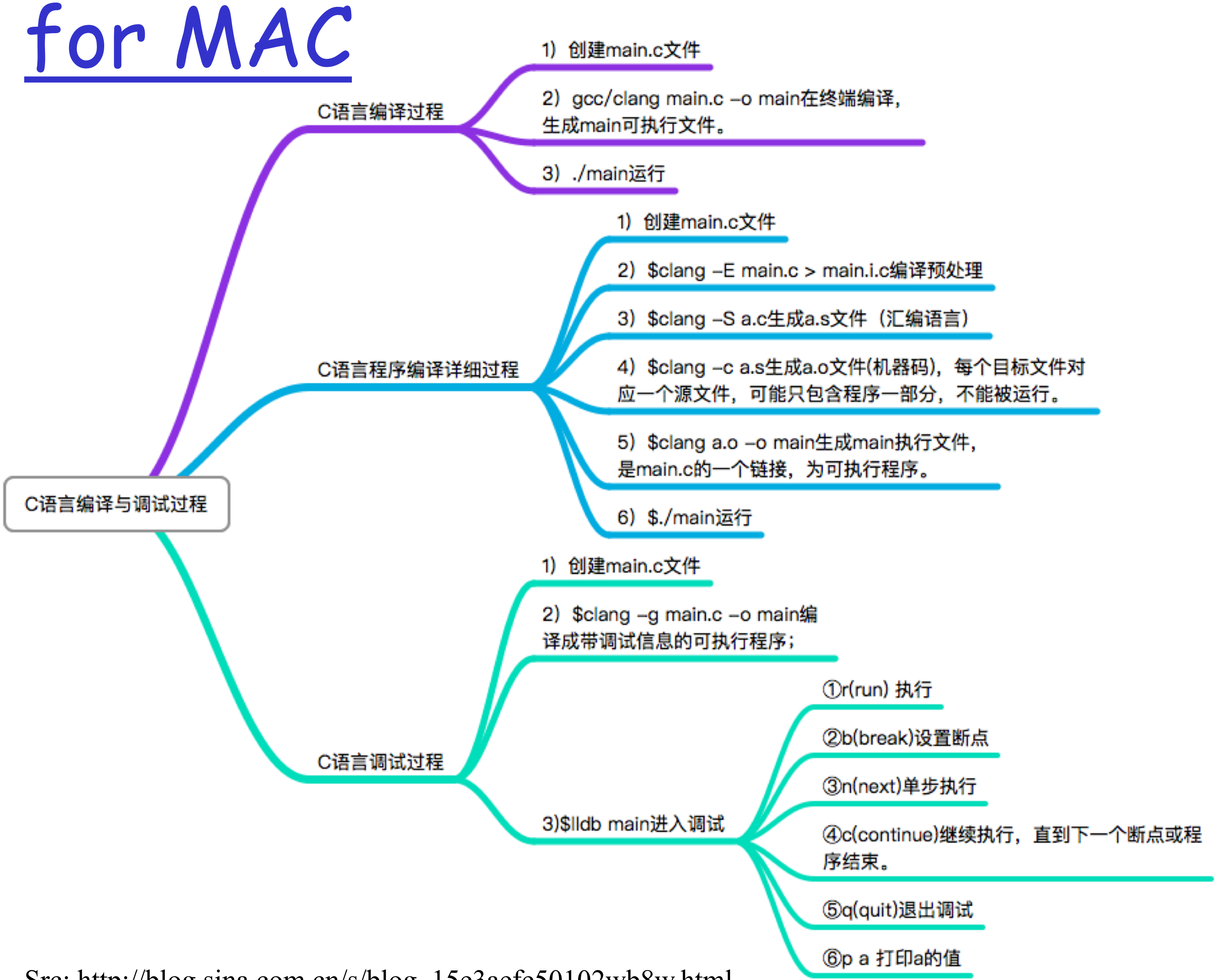
Backend

Machine Code

# Development & Debug Tools

- **GDB (The GNU Project Debugger)**
  - The program being debugged can be written in Ada, C, C++, Objective-C, Pascal (and many other languages).
  - http://www.gnu.org/software/gdb/
- **LLDB (Low Level Debugger)**
  - LLDB project builds on libraries provided by LLVM and Clang to provide a great native debugger.
  - **It is also blazing fast and much more memory efficient than GDB at loading symbols.**
  - http://lldb.llvm.org
- **GDB TO LLDB COMMAND MAP**
  - http://lldb.llvm.org/lldb-gdb.html

# for MAC

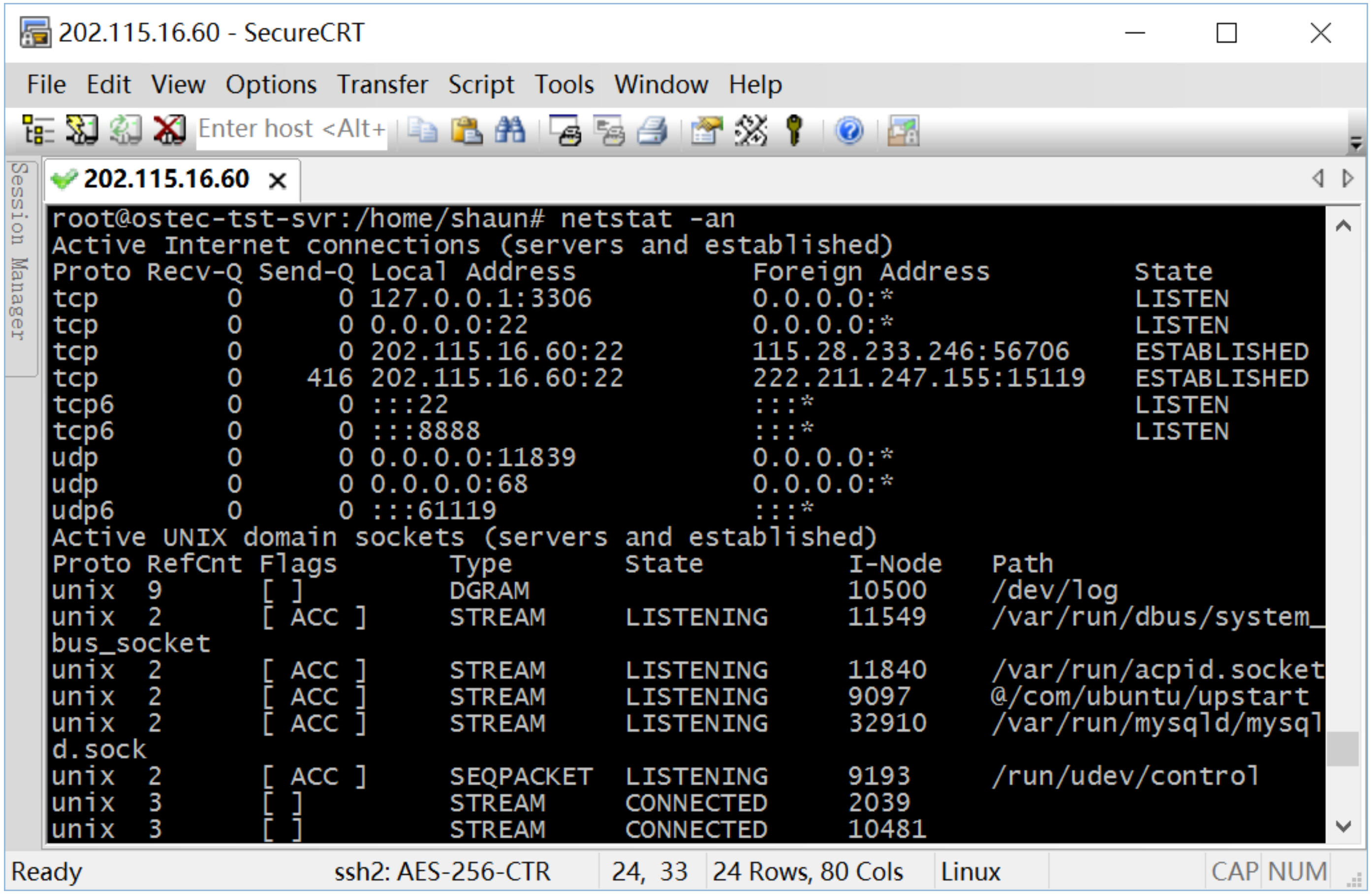


- C语言编译与调试过程
  - C语言编译过程
    - 1）创建main.c文件
    - 2）gcc/clang main.c –o main在终端编译，生成main可执行文件。
    - 3）./main运行
  - C语言程序编译详细过程
    - 1）创建main.c文件
    - 2）$clang –E main.c > main.i.c编译预处理
    - 3）$clang –S a.c生成a.s文件（汇编语言）
    - 4）$clang –c a.s生成a.o文件(机器码)，每个目标文件对应一个源文件，可能只包含程序一部分，不能被运行。
    - 5）$clang a.o –o main生成main执行文件，是main.c的一个链接，为可执行程序。
    - 6）$./main运行
  - C语言调试过程
    - 1）创建main.c文件
    - 2）$clang –g main.c –o main编译成带调试信息的可执行程序；
    - 3)$lldb main进入调试
      - ①r(run) 执行
      - ②b(break)设置断点
      - ③n(next)单步执行
      - ④c(continue)继续执行，直到下一个断点或程序结束。
      - ⑤q(quit)退出调试
      - ⑥p a 打印a的值

Src: http://blog.sina.com.cn/s/blog_15e3aefe50102wb8w.html

# Development & Debug Tools

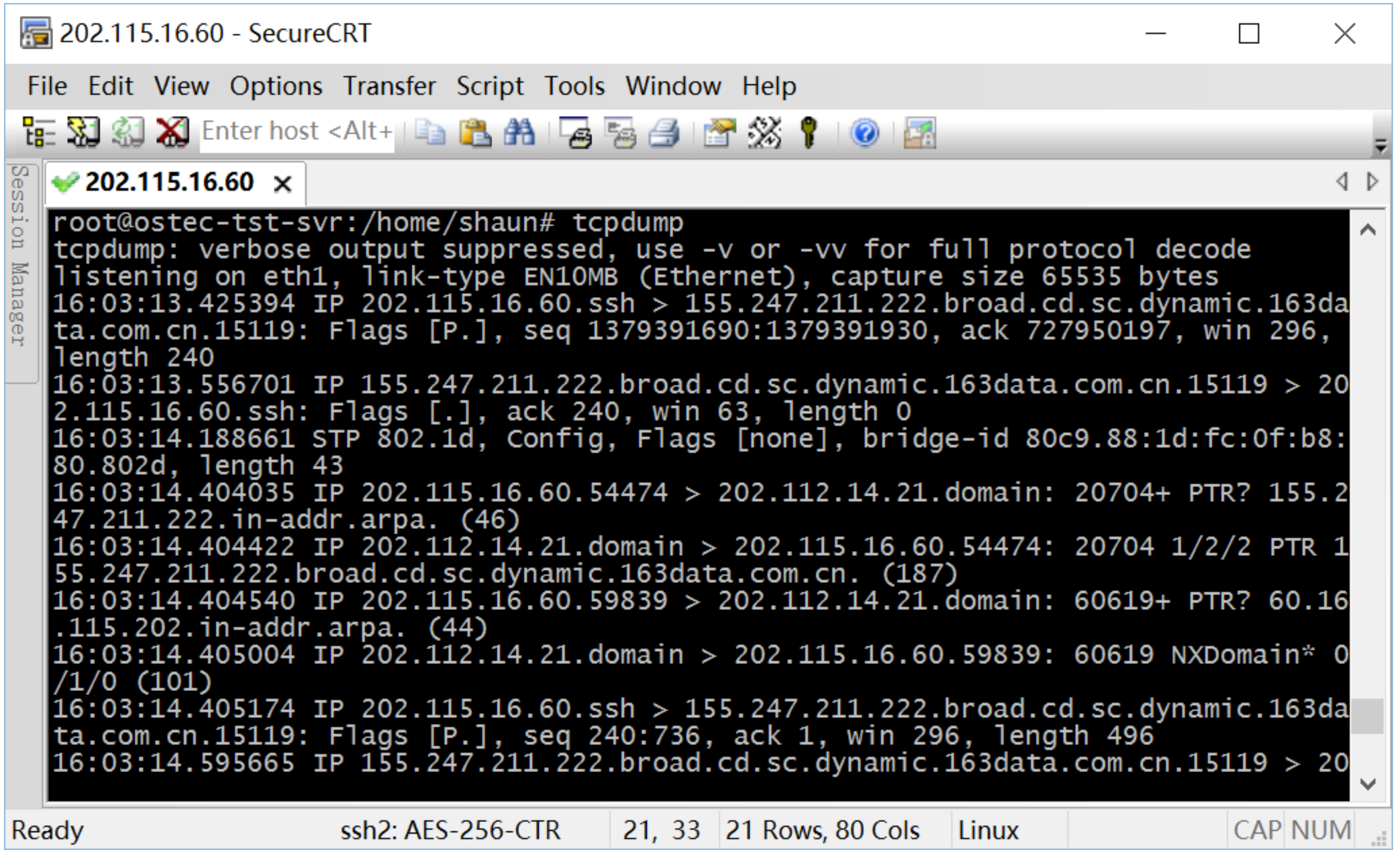
- ❑ netstat



```
root@ostec-tst-svr:/home/shaun# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 202.115.16.60:22        115.28.233.246:56706    ESTABLISHED
tcp        0    416 202.115.16.60:22        222.211.247.155:15119   ESTABLISHED
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::8888                 :::*                    LISTEN
udp        0      0 0.0.0.0:11839           0.0.0.0:*
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp6       0      0 :::61119                :::*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  9      [ ]         DGRAM                    10500    /dev/log
unix  2      [ ACC ]     STREAM     LISTENING     11549    /var/run/dbus/system_
bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     11840    /var/run/acpid.socket
unix  2      [ ACC ]     STREAM     LISTENING     9097     @/com/ubuntu/upstart
unix  2      [ ACC ]     STREAM     LISTENING     32910    /var/run/mysqld/mysql
d.sock
unix  2      [ ACC ]     SEQPACKET  LISTENING     9193     /run/udev/control
unix  3      [ ]         STREAM     CONNECTED     2039
unix  3      [ ]         STREAM     CONNECTED     10481
```

# Development & Debug Tools

- tcpdump



```
root@ostec-tst-svr:/home/shaun# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
16:03:13.425394 IP 202.115.16.60.ssh > 155.247.211.222.broad.cd.sc.dynamic.163da
ta.com.cn.15119: Flags [P.], seq 1379391690:1379391930, ack 727950197, win 296,
length 240
16:03:13.556701 IP 155.247.211.222.broad.cd.sc.dynamic.163data.com.cn.15119 > 20
2.115.16.60.ssh: Flags [.], ack 240, win 63, length 0
16:03:14.188661 STP 802.1d, Config, Flags [none], bridge-id 80c9.88:1d:fc:0f:b8:
80.802d, length 43
16:03:14.404035 IP 202.115.16.60.54474 > 202.112.14.21.domain: 20704+ PTR? 155.2
47.211.222.in-addr.arpa. (46)
16:03:14.404422 IP 202.112.14.21.domain > 202.115.16.60.54474: 20704 1/2/2 PTR 1
55.247.211.222.broad.cd.sc.dynamic.163data.com.cn. (187)
16:03:14.404540 IP 202.115.16.60.59839 > 202.112.14.21.domain: 60619+ PTR? 60.16
.115.202.in-addr.arpa. (44)
16:03:14.405004 IP 202.112.14.21.domain > 202.115.16.60.59839: 60619 NXDomain* 0
/1/0 (101)
16:03:14.405174 IP 202.115.16.60.ssh > 155.247.211.222.broad.cd.sc.dynamic.163da
ta.com.cn.15119: Flags [P.], seq 240:736, ack 1, win 296, length 496
16:03:14.595665 IP 155.247.211.222.broad.cd.sc.dynamic.163data.com.cn.15119 > 20
```