信息安全基础综合设计实验

Lecture 02

李经纬

电子科技大学

补充材料

Linux系统配置

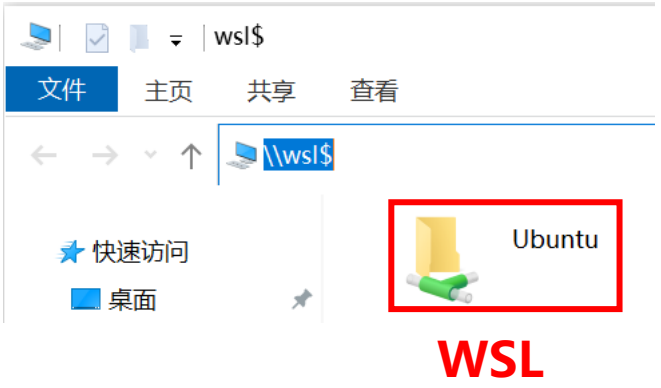
- **更换国内软件源**：阿里云、清华、中科大、163.....
  - 更换源后，执行`sudo apt-get update`更新软件包索引，以及`sudo apt-get upgrade`升级现有软件包
    - `sudo`为当前命令提升权限，要求输入口令，**口令默认不可见**

- 配置编程环境（更多**Linux基础**、**Vim简单使用**、**Bash脚本**）：

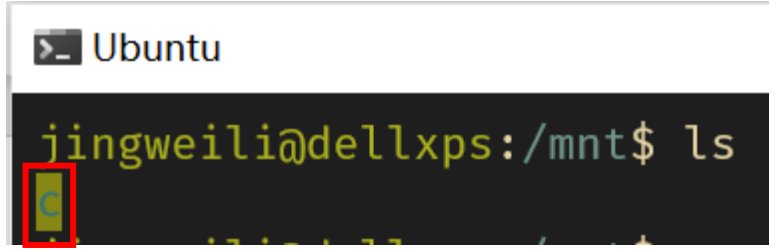
```
# 安装vim和编译工具
$ sudo apt-get install git cmake vim build-essential

# 配置vimrc
$ git clone -depth=1 https://github.com/amix/vimrc.git ~/.vim_runtime
$ sh ~/.vim_runtime/install_basic_vimrc.sh
```

与宿主操作系统互访问

- 宿主操作系统访问WSL：**\\wsl$**
  - 可通过**<win+r>**运行窗口或资源管理器地址栏访问
  - 用户目录位于“**\home\{用户名}**”
    - 可以将作业包解压至WSL的“**\home\{用户名}**”目录



- WSL访问宿主操作系统：所有盘符挂载在“**/mnt**”目录下
  - 勿在WSL中直接修改宿主操作系统文件



- Vmware互访问依赖VM Tools

课程回顾

Linux简单编程

- VIM编辑器
  - 三种模式切换；hjkl光标移动
- 编译指令：`g++ -o <可执行文件名> <源文件名>`
- Makefile
  - 由一系列**规则**构成
  - 每条规则包含**目标、依赖、指令**

```
TARGET: PREREQUISITE ...
        COMMAND ...
        COMMAND ...
```

进制转换

- 二进制转十进制
  - 思路：逐字符判断，若为“1”则偏移一位（乘以2），否则不进位
  - 代码参考

```
// 从左到右用二进制的每个数去乘以2的相应次方并递增
for(int i = 0; i < input.length(); ++i) {
    if(input[i] == '1') {
        ++value;
    }
    value = value << 1;
}
return value >> 1; // 去除多余的移位操作
```

进制转换

- 十进制转二进制
  - 思路：除二取余，然后倒序排列
  - 代码参考

```
while(input > 0) {
    if(input % 2 == 1) {
        binary.insert(0, "1");
    } else {
        binary.insert(0, "0");
    }
    input = input >> 1; // 取除以2后的数
}
return binary;
```

数论基础实验 —— 模指数运算

模指数运算

- 目标：已知正整数a、e、m，计算**$a^e \bmod m$**
  - a：底数；e：指数；m：模数
  - 应用：RSA密码

- 问题：计算$2^{90} \bmod 13$

```
double pow (double base, double exponent);
```

返回**base**的**expoent**次幂

头文件：**math.h**

其他几种pow形式，参考网址

溢出的代码

```
#include <stdio.h>
#include <math.h>

int mod_exp(int base, int exp, int mod) {
    return (int) pow(base, exp) % mod;
}

int main() {
    printf("%d\n", mod_exp(2, 90, 13));
    return 0;
}
```

溢出的代码

```
#include <stdio.h>
#include <math.h>

int mod_exp(int base, int exp, int mod) {
    return (int) pow(base, exp) % mod;
}

int main() {
    printf("%d\n", mod_exp(2, 90, 13));
    return 0;
}
```

当**base**和**exp**足够大时，**pow**返回值超出**int**所能表示的最大范围，发生**溢出**

溢出的代码

```
#include <stdio.h>
#include <math.h>

int mod_exp(int base, int exp, int mod) {
    return (int) pow(base, exp) % mod;
}

int main() {
    printf("%d\n", mod_exp(2, 90, 13));
    return 0;
}
```

当**base**和**exp**足够大时，**pow**返回值超出**int**所能表示的最大范围，发生**溢出**

问题：如何有效计算模指数

分治原理

- 分治法：**分→治→归并**
  - $a^e \bmod m = a^{e1+e2} \bmod m = (a^{e1} \bmod m) * (a^{e2} \bmod m) \bmod m$
- 示例：计算$2^{90} \bmod 13$
  - **分**：$2^{90} = 2^{50} * 2^{40}$
  - **治**：$2^{40} \bmod 13 = 3$，$2^{50} \bmod 13 = 4$
  - **归并**：$2^{90} \bmod 13 = 4 * 3 \bmod 13 = 12$

2^{40}和2^{50}数值较小，不会产生溢出

分治原理

- 分治法：**分→治→归并**
  - $a^e \bmod m = a^{e1+e2} \bmod m = (a^{e1} \bmod m) * (a^{e2} \bmod m) \bmod m$
- 示例：计算$2^{90} \bmod 13$
  - **分**：$2^{90} = 2^{50} * 2^{40}$
  - **治**：$2^{40} \bmod 13 = 3$，$2^{50} \bmod 13 = 4$
  - **归并**：$2^{90} \bmod 13 = 4 * 3 \bmod 13 = 12$
- **分拆规则？**

2^{40}和2^{50}数值较小，不会产生溢出

二进制算法

- 以**二进制**方式分拆指数
  - $e = d_{n-1} * 2^{n-1} + \ldots + d_1 * 2 + d_0 = D_{n-1} + \ldots + D_1 + D_0$
  - $a^e = a^{Dn-1} * a^{Dn-2} * \ldots * a^{D1} * a^{D0}$
- 分治计算: $A_i = a^{Di} \bmod m = a^{2^i} \bmod m$ 或 1
  - $a^{2^{i+1}} \bmod m = (a^{2^i} \bmod m) * (a^{2^i} \bmod m) \bmod m$
  - 计算**$O(\log_2 e)$**次
- 归并: $(A_{n-1} * \ldots * A_0) \bmod m$

模指数运算示例 I：7^{256} mod 13

- **分拆指数：**256 = 2^8
- **分治计算：**
  - 7^1 mod 13 = 7
  - 7^2 mod 13 = [(7^1 mod 13) * (7^1 mod 13)] mod 13 = 10
  - 7^4 mod 13 = 9 , 7^8 mod 13 = 3 , …
  - 7^{256} mod 13 = [(7^{128} mod 13) * (7^{128} mod 13)] mod 13 = 9
- **归并：**7^{256} **mod** 13 = 9

模指数运算示例 II：$5^{117} \bmod 19$

- **分拆指数：** $117 = 1 + 4 + 16 + 32 + 64$
- **分治计算：**
  - $5^1 \bmod 19 = 5$
  - $5^2 \bmod 19 = [(5^1 \bmod 19) * (5^1 \bmod 19)] \bmod 19 = 6$
  - $5^4 \bmod 19 = 17$; … $5^{16} \bmod 19 = 16$; … $5^{32} \bmod 19 = 9$
  - $5^{64} \bmod 19 = [(5^{32} \bmod 19) * (5^{32} \bmod 19)] \bmod 19 = 5$
- **归并：** $5^{117} \bmod 19 = 5^{1+4+16+32+64} \bmod 19 = 1$

数论基础实验 —— 素性测试

素数

- 如果n (n > 1) 是素数，当且仅当n只有因子1和它本身
  - 注：1不是素数
- 数学应用：任意自然数可由全体素数基底表达
- 素性测试：**判定n是否为素数**

素性判定基本方法

- 方法 I：判断n是否被i整除 (i = 2, 3,..., n-1)
  - 素数n只有1和n两个因子

素性判定基本方法

- 方法 I：判断n是否被i整除 (i = 2, 3,..., n-1)
  - 素数n只有1和n两个因子
  - n-2个判断条件，**如何减少判断次数？**

素性判定基本方法

- 方法 I：判断n是否被i整除 (i = 2, 3,..., n-1)
  - 素数n只有1和n两个因子
  - n-2个判断条件，**如何减少判断次数？**
- 方法II：判断n是否被i整除 (i = 2, 3,..., $n^{1/2}$)

素性判定基本方法

- 方法 I：判断n是否被i整除 (i = 2, 3,..., n-1)
  - 素数n只有1和n两个因子
  - n-2个判断条件，**如何减少判断次数？**
- 方法II：判断n是否被i整除 (i = 2, 3,..., $n^{1/2}$)
  - 如果j > $n^{1/2}$是n的因子，总能找到i < $n^{1/2}$，满足n = i * j
  - 判断次数减少为$n^{1/2}$ -1次，复杂度由O(n)降低为O($n^{1/2}$)

素性判定基本方法

- 方法 I：判断n是否被i整除 (i = 2, 3,..., n-1)
  - 素数n只有1和n两个因子
  - n-2个判断条件，**如何减少判断次数？**
- 方法II：判断n是否被i整除 (i = 2, 3,..., $n^{1/2}$)
  - 如果j > $n^{1/2}$是n的因子，总能找到i < $n^{1/2}$，满足n = i * j
  - 判断次数减少为$n^{1/2}$ -1次，复杂度由O(n)降低为O($n^{1/2}$)
  - **如何进一步减少判断次数？**

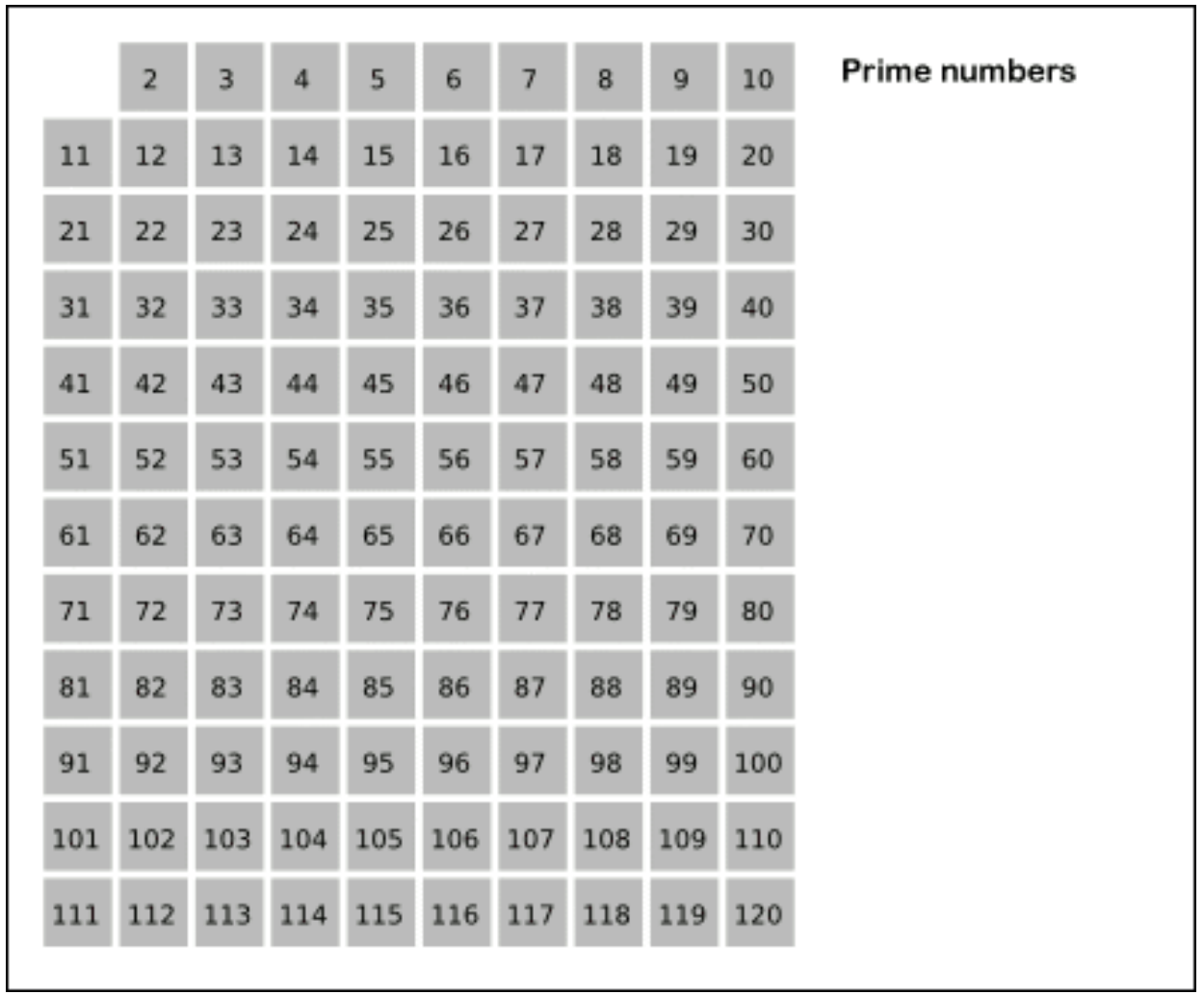
一种确定性素性判定思想

- **确定性**：判断n**一定**是/不是素数
- 思想：**找出{2, 3,..., $n^{1/2}$}中的素数，判断n是否含有这些素因子**
  - {2, 3,..., $n^{1/2}$}可能存在合数，合数含有素因子导致重复判断
- 步骤：（1）确定待筛选集合{2, 3,..., $n^{1/2}$}；（2）基于{2, 3,..., $n^{1/2}$}的**Eratosthenes筛选**；（3）筛选后元素与n整除判定

Eratosthenes筛选法

- 目标：**产生最小N个素数（例如$N=n^{1/2}$）**
  - 不适用于计算某个范围内全部素数
- 取**第一个素数**2, 划去{2,…, N}中除2以外所有2的倍数
- 大于2的第一个正整数(即3)被认定为素数, 在余下的整数中划去除3以外所有3的倍数
- 循环此过程直到找到{2, 3,…, N}中的所有素数

Eratosthenes筛选法示例



基于筛选法素性测试

判断2543是否为素数

- **求平方根**：$2543^{1/2}$ ~= 50，确定待筛选集合{2, 3, 4,..., 50}
- 使用**Eratosthenes筛选法**在集合{2, 3, 4,..., 50}中筛选出所有素数: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47
- **整除判定**：2543除以这15个素数，结果都不是整数，因此2543**一定**是素数

课程作业

作业要求

- 课堂完成预留作业：助教同学检查通过+登记
  - 在线情况下，私聊检查程序**测试结果、源代码**

作业要求

截止时间：2023年9月21日00:00

- **步骤1**：在压缩包内`implementation/unit_x.cc`完成相关函数的实现（函数头已在源文件内定义）
- **步骤2**：根据压缩包内`README.pdf`指示，完成程序测试
- **步骤3**：发送邮件至**maplemeo@proton.me**：
  - 邮件题目：unit_x-姓名-学号
  - 包含附件：unit_x.cc（函数实现源文件）和test.png（测试结果截图）
  - 请勿包含任何正文内容

作业要求

截止时间：2023年9月21日00:00

- **步骤1**：在压缩包内`implementation/unit_x.cc`完成相关函数的实现（函数头已在源文件内定义）
- **步骤2**：根据压缩包内`README.pdf`指示，完成程序测试
- **步骤3**：发送邮件至**maplemeo@proton.me**：
  - 邮件题目：unit_x-姓名-学号
  - 包含附件：unit_x.cc（函数实现源文件）和test.png（测试结果截图）
  - 请勿包含任何正文内容

正确完成课程作业，获得平时成绩加分

模指数运算

- 实现模指数运算的二进制算法
- 函数头：

```
unsigned int mod_exp(unsigned int a, \
                     unsigned int e, \
                     unsigned int m);
// a: 输入底数
// e: 输入指数
// m: 输入模数
// 返回: a^e mod m
```

素性测试 I

- 实现基于Eratosthenes筛选的素性测试算法
- 函数头：

```
bool prime_test(unsigned int a);
// a: 输入测试正整数
// 返回: true如果a为素数；false如果a不为素数
```