北京科技大学实验报告

学院： 计算机与通信工程学院　　专业： 计算机科学与技术　　班级： 计 184

姓名： 王丹琳　　学号： 41824179　　实验日期： 2021 年 4 月 22 日

实验名称：

实验五 Linux 环境下的 C 编程 24 点游戏

实验目的：

Unix/Linux 系统的 C 编译器为 gcc，它是 GNU 推出的功能强大的编译工具，因为 UNIX 系统的编译器为 cc，所以在 Linux 系统中还保留一个链接 cc 用于和 UNIX 的向后兼容。Linux 的面向对象的 C 编译器为 g++或 c++，它不仅功能强大、结构灵活，且可以通过不同的可选模块来支持多种语言，如 Java、Fortran77、Ada 和汇编等。gcc/g++能将 C/C++源程序和目标程序编译并调用链接程序 ld 生成可执行文件，如果没有指定可执行文件的名字，则默认生成 a.out。

1）了解 C 编程在 Linux 系统中的运用。

2）练习 Linux 系统的 C 编程，掌握 Linux 系统 C 编程的基本语法，能够使用 C 在 Linux 系统编写简单程序，并调试运行。

实验仪器：

PC 机一台：ThinkPad T480

实验环境：

VMware 虚拟机

Linux Ubuntu 64 位

实验原理：

Linux，全称 GNU/Linux，是一种免费使用和自由传播的类 UNIX 操作系统，其内核由林纳斯·本纳第克特·托瓦兹于 1991 年 10 月 5 日首次发布，它主要受到 Minix 和 Unix 思想的启发，是一个基于 POSIX 的多用户、多任务、支持多线程和多 CPU 的操作系统。它能运行主要的 Unix 工具软件、应用程序和网络协议。它支持 32 位和 64 位硬件。Linux 继承了 Unix 以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。

实验内容与步骤：

1） 根据 gcc 的相关语法和功能编写 24 点游戏，注意设计好主函数（类）、子函数（子类）及相关头文件，明确各自功能及调用关系。

2） 参照本讲 Autotools 示例，使用 Autotools 工具生成 Makefile 文件，完成源码安装，并制作源码安装包。

3） 有条件的小伙伴们，请安装(GTK+/Glade/Anjuta/Qt）图形界面开发工具之一，为上述 24 点游戏程序，增加设计友好的图形界面 GUI，编译生成可发布的执行文件。

实验数据及处理：

源代码：

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
char op[4]={'+', '-', '*', '/',};
double computeTwo(double A, int op, double B) {
  switch(op) {
  case 0 : return A + B;
  case 1 : return A - B;
  case 2 : return A * B;
  case 3 : if (B != 0) {
  return A / B;
  } else {
  return 255;
  }
  default : return 0;
  }
}
int output (int i, int j, int k, int A, int B, int C, int D) {
  //printf("%d%c%d%c%d%c%d\n", A, op[i], B, op[j], C, op[k], D);
```

```
  int solutionNum = 0;
  double result1 = computeTwo(computeTwo(computeTwo(A, i, B), j, C), k,
D);
  double result2 = computeTwo(computeTwo(A, i, computeTwo(B, j, C)), k,
D);
  double result3 = computeTwo(A, i, computeTwo(B, j, computeTwo(C, k, D)));
  double result4 = computeTwo(A, i, computeTwo(computeTwo(B, j, C), k, D));
  double result5 = computeTwo(computeTwo(A, i, B), j, computeTwo(C, k, D));
if (fabs(result1 - 24) < 1e-6) {
  printf("((%d%c%d)%c%d)%c%d = 24\n", A, op[i], B, op[j], C, op[k], D);
  solutionNum += 1;
  }
  if (fabs(result2 - 24) < 1e-6) {
  printf("(%d%c(%d%c%d))%c%d = 24\n", A, op[i], B, op[j], C, op[k], D);
  solutionNum += 1;
  }
  if (fabs(result3 - 24) < 1e-6) {
  printf("%d%c(%d%c(%d%c%d)) = 24\n", A, op[i], B, op[j], C, op[k], D);
  solutionNum += 1;
  }
  if (fabs(result4 - 24) < 1e-6) {
  printf("%d%c((%d%c%d)%c%d) = 24\n", A, op[i], B, op[j], C, op[k], D);
  solutionNum += 1;
  }
  if (fabs(result5 - 24) < 1e-6) {
  printf("(%d%c%d)%c(%d%c%d) = 24\n", A, op[i], B, op[j], C, op[k], D);
  solutionNum += 1;
  }
  //printf("Result:%f %f %f %f %f\n", result1, result2, result3, result4,
result5);
```

```
  return solutionNum;
}
int computeFour4(int A, int B, int C, int D) {
  int solutionNum = 0;
  for(int i = 0; i < 4; i += 1) {
  for(int j = 0; j < 4; j += 1) {
  for(int k = 0; k < 4; k += 1) {
  solutionNum += output(i, j, k, A, B, C, D);
  solutionNum += output(i, j, k, A, B, D, C);
  solutionNum += output(i, j, k, A, C, B, D);
  solutionNum += output(i, j, k, A, C, D, B);
  solutionNum += output(i, j, k, A, D, B, C);
  solutionNum += output(i, j, k, A, D, C, B);
  solutionNum += output(i, j, k, B, A, C, D);
  solutionNum += output(i, j, k, B, A, D, C);
  solutionNum += output(i, j, k, B, C, A, D);
  solutionNum += output(i, j, k, B, C, D, A);
  solutionNum += output(i, j, k, B, D, A, C);
  solutionNum += output(i, j, k, B, D, C, A);
  solutionNum += output(i, j, k, C, A, B, D);
solutionNum += output(i, j, k, C, A, D, B);
  solutionNum += output(i, j, k, C, B, A, D);
  solutionNum += output(i, j, k, C, B, D, A);
  solutionNum += output(i, j, k, C, D, A, B);
  solutionNum += output(i, j, k, C, D, B, A);
  solutionNum += output(i, j, k, D, A, B, C);
  solutionNum += output(i, j, k, D, A, C, B);
  solutionNum += output(i, j, k, D, B, A, C);
  solutionNum += output(i, j, k, D, B, C, A);
  solutionNum += output(i, j, k, D, C, A, B);
```

```
  solutionNum += output(i, j, k, D, C, B, A);
  }
  }
  }
  return solutionNum;
}
int computeFourX(int A, int B, int C, int D) {
  int solutionNum = 0;
  for(int i = 0; i < 4; i += 1) {
  for(int j = 0; j < 4; j += 1) {
  for(int k = 0; k < 4; k += 1) {
  solutionNum += output(i, j, k, A, A, C, C);
  solutionNum += output(i, j, k, A, C, A, C);
  solutionNum += output(i, j, k, A, C, C, A);
  solutionNum += output(i, j, k, C, A, A, C);
  solutionNum += output(i, j, k, C, A, C, A);
  solutionNum += output(i, j, k, C, C, A, A);
  }
  }
  }
  return solutionNum;
}
int computeFour3(int A, int B, int C, int D) {
  int solutionNum = 0;
  for(int i = 0; i < 4; i += 1) {
  for(int j = 0; j < 4; j += 1) {
  for(int k = 0; k < 4; k += 1) {
  solutionNum += output(i, j, k, A, A, C, D);
  solutionNum += output(i, j, k, A, A, D, C);
  solutionNum += output(i, j, k, A, C, A, D);
```

```
solutionNum += output(i, j, k, A, C, D, A);
  solutionNum += output(i, j, k, A, D, A, C);
  solutionNum += output(i, j, k, A, D, C, A);
  solutionNum += output(i, j, k, C, A, A, D);
  solutionNum += output(i, j, k, C, A, D, A);
  solutionNum += output(i, j, k, C, D, A, A);
  solutionNum += output(i, j, k, D, A, A, C);
  solutionNum += output(i, j, k, D, A, C, A);
  solutionNum += output(i, j, k, D, C, A, A);
  }
  }
  }
  return solutionNum;
}
int computeFour2(int A, int B, int C, int D) {
  int solutionNum = 0;
  for(int i = 0; i < 4; i += 1) {
  for(int j = 0; j < 4; j += 1) {
  for(int k = 0; k < 4; k += 1) {
  solutionNum += output(i, j, k, A, A, A, D);
  solutionNum += output(i, j, k, A, A, D, A);
  solutionNum += output(i, j, k, A, D, A, A);
  solutionNum += output(i, j, k, D, A, A, A);
  }
  }
  }
  return solutionNum;
}
int computeFour1(int A, int B, int C, int D) {
  int solutionNum = 0;
```

```
  for(int i = 0; i < 4; i += 1) {
  for(int j = 0; j < 4; j += 1) {
  for(int k = 0; k < 4; k += 1) {
  solutionNum += output(i, j, k, A, A, A, A);
  }
  }
  }
  return solutionNum;
}
int comp(const void*a, const void*b) {
  return *(int*)b - *(int*)a;
}
int main() {
  int num[4];
  int result = 0;
  printf("Please input 4 integers:\n");
  scanf("%d %d %d %d",num,num + 1,num + 2,num + 3);
  if (num[0] == num[1] && num[1] == num[2] && num[2] == num[3]) {
  result = computeFour1(num[0], num[1], num[2], num[3]);
  } else {
  qsort(num, 4, sizeof(int), comp);
  if (num[0] == num[1] && num[1] == num[2]) {
  result = computeFour2(num[0], num[1], num[2], num[3]);
  } else if (num[0] == num[1] && num[2] == num[3]) {
  result = computeFourX(num[0], num[1], num[2], num[3]);
  } else if (num[0] == num[1]) {
  result = computeFour3(num[0], num[1], num[2], num[3]);
  } else {
  result = computeFour4(num[0], num[1], num[2], num[3]);
  }
```

```
 }
 printf("%d Solutions.", result);
 return 0;
}
```

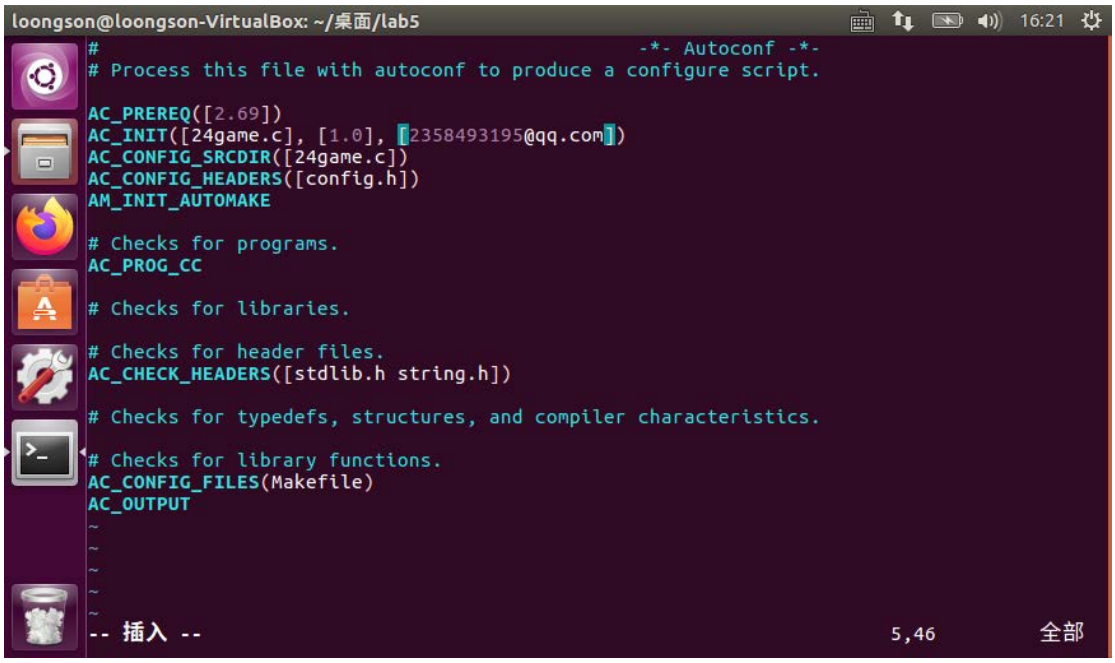
2. 使用 Autotools 工具生成 Makefile 文件，完成源码安装

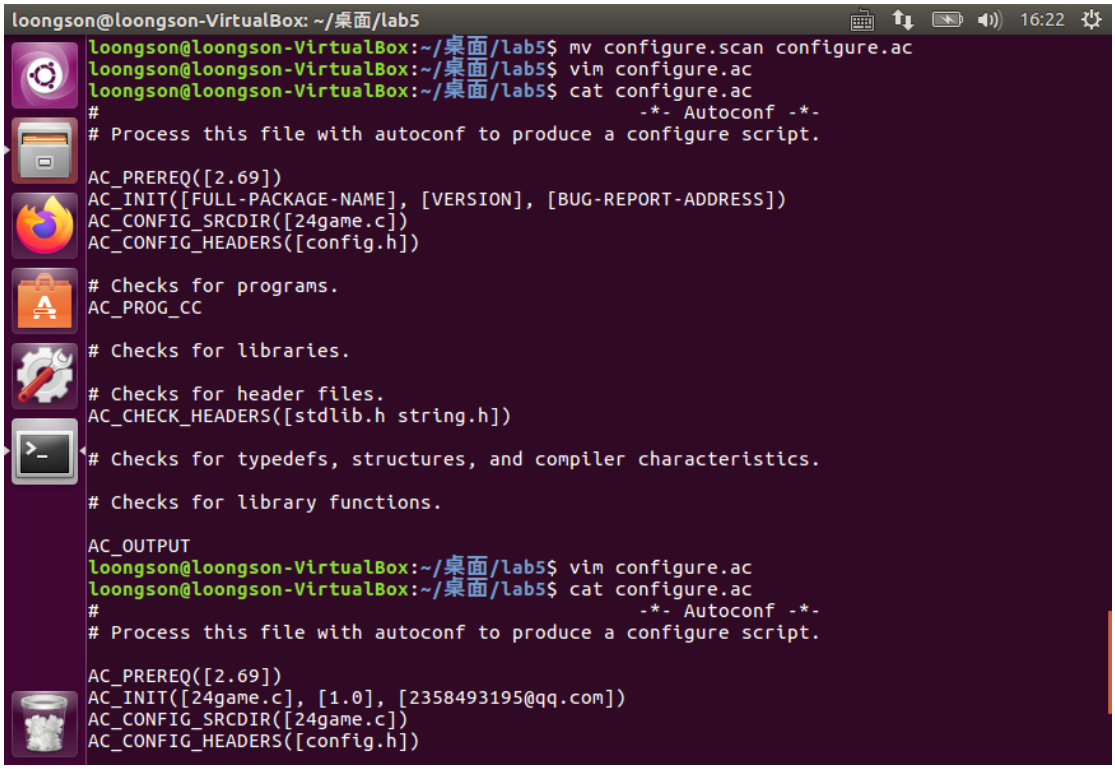
```
loongson@loongson-VirtualBox:~/桌面/lab5$ cat 24game.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
char op[4]={'+', '-', '*', '/',};
double computeTwo(double A, int op, double B) {
 switch(op) {
 case 0 : return A + B;
 case 1 : return A - B;
 case 2 : return A * B;
 case 3 : if (B != 0) {
 return A / B;
 } else {
 return 255;
 }
 default : return 0;
 }
}
int output (int i, int j, int k, int A, int B, int C, int D) {
 //printf("%d%c%d%c%d%c%d\n", A, op[i], B, op[j], C, op[k], D);
 int solutionNum = 0;
 double result1 = computeTwo(computeTwo(computeTwo(A, i, B), j, C), k,
D);
 double result2 = computeTwo(computeTwo(A, i, computeTwo(B, [illegible] k,
D);
 double result3 = computeTwo(A, i, computeTwo(B, j, computeTwo(C, k, D)));
 double result4 = computeTwo(A, i, computeTwo(computeTwo(B, j, C), k, D));
```

执行命令 autoscan

```
loongson@loongson-VirtualBox:~/桌面/lab5$ ls
24game.c  autoscan.log  configure.scan
loongson@loongson-VirtualBox:~/桌面/lab5$
```

将文件 configure.scan 重命名为 configure.ac，然后再编辑修改这个配置文件（增加 AM_INIT_AUTOMAKE 字段、AC_CONFIG_FILES(Makefile)字段、修改 AC_INIT 字段）：





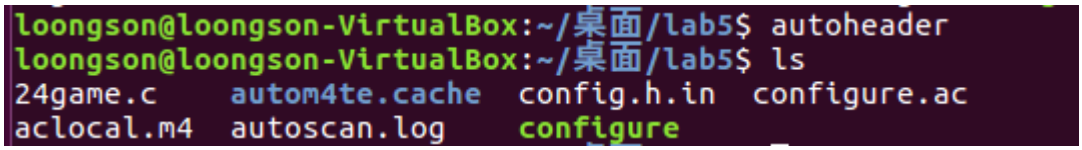
在项目目录下执行 aclocal 命令，扫描 configure.ac 文件生成 aclocal.m4 文件：

```
loongson@loongson-VirtualBox:~/桌面/lab5$ aclocal
loongson@loongson-VirtualBox:~/桌面/lab5$ ls
24game.c  aclocal.m4  autom4te.cache  autoscan.log  configure.ac
```

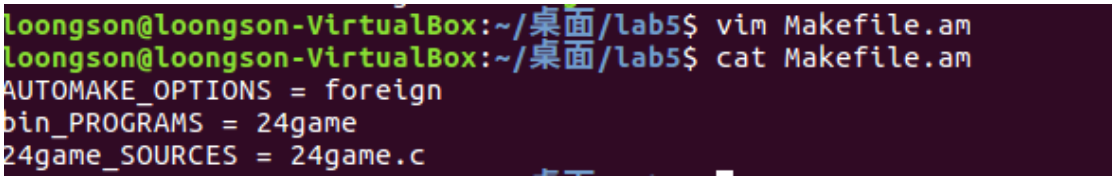
在项目目录下执行 autoconf 命令生成 configure 文件：

```
loongson@loongson-VirtualBox:~/桌面/lab5$ autoconf
loongson@loongson-VirtualBox:~/桌面/lab5$ ls
24game.c  aclocal.m4  autom4te.cache  autoscan.log  configure  configure.ac
```

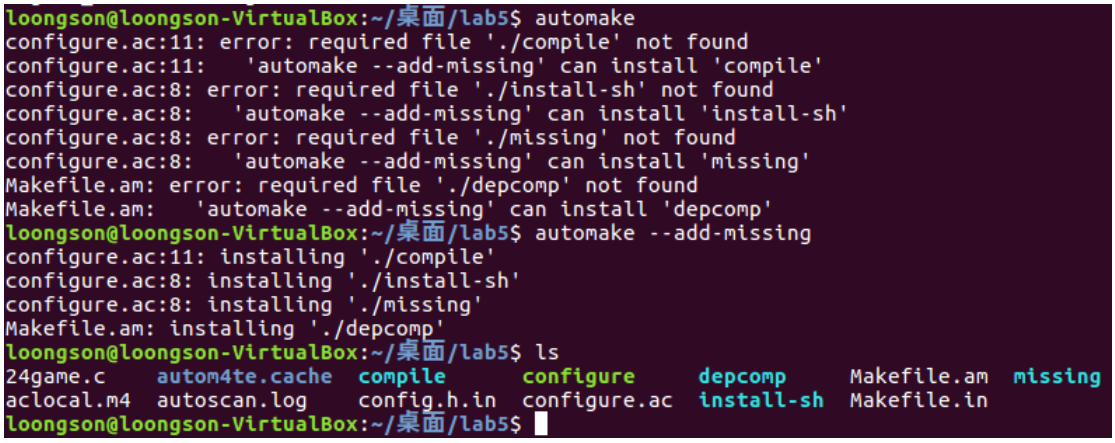
在项目目录下执行 autoheader 命令生成 config.h.in 文件：



在项目目录下创建一个 Makefile.am 文件，供 automake 工具根据 configure.in 中的参数将 Makefile.am 转换成 Makefile.in 文件



在项目目录下执行 automake 命令生成 Makefile.in 文件（经测试发现应使用选项--add-missing 让 automake 自动添加必需的脚本文件）：



在项目目录下执行./configure 命令，基于 Makefile.in 生成最终的 Makefile 文件。该命令将一些配置参数添加到 Makefile 文件中:

```
loongson@loongson-VirtualBox:~/桌面/lab5$ ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /bin/grep
checking for egrep... /bin/grep -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
```

在项目目录下执行 make 命令，基于 Makefile 文件编译源代码文件并生成可执行文件：

```
loongson@loongson-VirtualBox:~/桌面/lab5$ make
make  all-am
make[1]: Entering directory '/home/loongson/桌面/lab5'
gcc -DHAVE_CONFIG_H -I.     -g -O2 -MT 24game.o -MD -MP -MF .deps/24game.Tpo -c -o 24gam
e.o 24game.c
24game.c: In function ‘output’:
```

在项目目录下执行 make install 命令将编译后的软件包安装到系统中

3.制作源码安装包

输入命令 make dist，生成一个.tar.gz 格式的源代码压缩包：

4.运行软件，输入 5 5 1 5，得到的结果如下

```
Please input 4 integers:
5 5 1 5
(5-(1/5))*5 = 24
5*(5-(1/5)) = 24
```

实验结果与分析：

通过本次实验，我对 Linux 环境下的 C 编程内容进行了复习和理解。通过练

习，掌握了 Linux 系统 C 编程的基本语法，能够使用 C 在 Linux 系统编写 24 点程序，并调试运行。同时，通过查询资料，使用 Autotools 工具生成 Makefile 文件，完成源码安装，并制作源码安装包。

Unix/Linux 系统的 C 编译器为 gcc，它是 GNU 推出的功能强大的编译工具，因为 UNIX 系统的编译器为 cc，所以在 Linux 系统中还保留一个链接 cc 用于和 UNIX 的向后兼容。Linux 的面向对象的 C 编译器为 g++或 c++，它不仅功能强大、结构灵活。gcc/g++能将 C/C++源程序和目标程序编译并调用链接程序 ld 生成可执行文件，如果没有指定可执行文件的名字，则默认生成 a.out。在实验中，我熟悉了 Linux 环境下的 C 编程，为今后的 Linux 使用打下扎实的基础。但要真正掌握，还要在实际操作中多去使用。