**凯里学院** Linux操作系统与原理应用 **实验报告**

装订线

装订线

数据科学与大数据技术 专业 2018 年级 学号2018402207 姓名 唐涛

成绩

合作者 石庆鹏、万志豪、何得胜、李佳 实验日期 2020 年 5 月 28 日

指导教师 李韦鹏 评阅日期 年 月 日

实验12 Linux软件设计

1. 实验目的

完成一件Linux环境下的软件作品，或研究分析一个Linux操作系统子模块源码，将本学期所学知识融入其中，达到课程训练和提升能力的目的，重点是通过实践突破同学们自身对Linux操作系统的原理认知实验项目

贪吃蛇小游戏

1. 实验要求

按照之前确定的组队情况，以小队为单位完成一款Linux软件的开发，鼓励借鉴网上成熟作品，并加上自己的理解和改进。

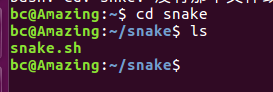
* 1. 需要提交的内容包括：
  2. 软件源码，以github公共仓的链接形势放在实验报告中
  3. 编译出的可执行文件，并对运行结果截图或者录制视频
  4. 一份描述合理的实验报告，重点描述软件运行原理和开发过程中解决问题的案例，要求一方面上传到github，一方面打印纸质版交给任课教师

1. 实验结果

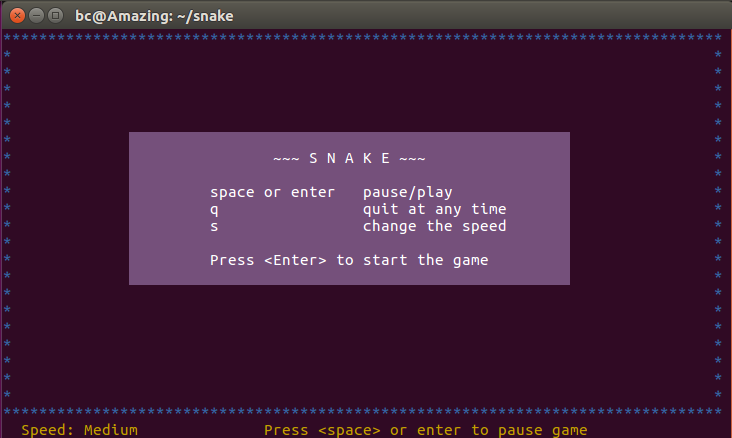
GitHub网址：<https://github.com/2018402207/Linux>

实验截图：

可执行文件：



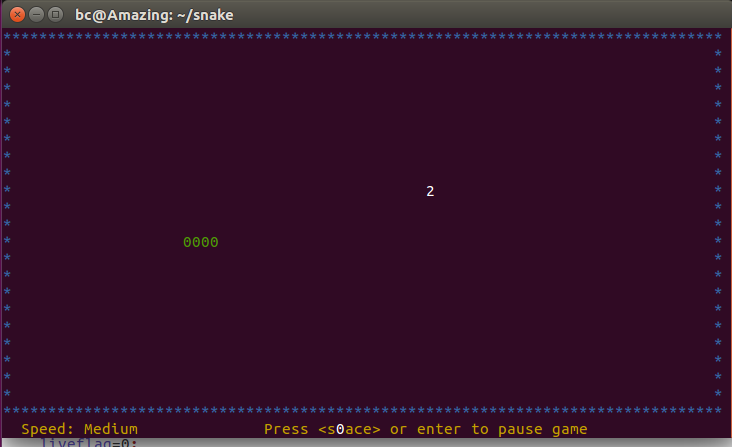
开始界面



结束界面



游戏界面



代码内容及分析：

#!/usr/bin/env bash ---shell脚本必备

# filename: snake.sh

# snake game

----定义游戏开始和结束界面的大小及提示的位置

good\_game=(

' '

' G A M E O V E R ! '

' '

' Score: '

' press q to quit '

' press n to start a new game '

' press s to change the speed '

' '

);

game\_start=(

' '

' ~~~ S N A K E ~~~ '

' '

' space or enter pause/play '

' q quit at any time '

' s change the speed '

' '

' Press <Enter> to start the game '

' '

);

snake\_exit() { #退出游戏

stty echo; #恢复回显

tput rmcup; #恢复屏幕

tput cvvis; #恢复光标

exit 0;

}

draw\_gui() { # 画边框

clear;

color="\033[34m\*\033[0m"; #背景颜色

for (( i = 0; i < $1; i++ )); do

echo -ne "\033[$i;0H${color}";

echo -ne "\033[$i;$2H${color}";

done

for (( i = 0; i <= $2; i++ )); do

echo -ne "\033[0;${i}H${color}";

echo -ne "\033[$1;${i}H${color}";

done

ch\_speed 0;

echo -ne "\033[$Lines;$((yscore-10))H\033[36mScores: 0\033[0m";

echo -en "\033[$Lines;$((Cols-50))H\033[33mPress <space> or enter to pause game\033[0m";

}

snake\_init() {

Lines=`tput lines`; Cols=`tput cols`; #得到屏幕的长宽

xline=$((Lines/2)); ycols=4; #开始的位置

xscore=$Lines; yscore=$((Cols/2)); #打印分数的位置

xcent=$xline; ycent=$yscore; #中心点位置

xrand=0; yrand=0; #随机点

sumscore=0; liveflag=1; #总分和点存在标记

sumnode=0; foodscore=0; #总共要加长的节点和点的分数

snake="0000 "; #初始化贪吃蛇

pos=(right right right right right); #开始节点的方向

xpt=($xline $xline $xline $xline $xline); #开始的各个节点的x坐标

ypt=(5 4 3 2 1); #开始的各个节点的y坐标

speed=(0.05 0.1 0.15); spk=${spk:-1}; #速度 默认速度

draw\_gui $((Lines-1)) $Cols

}

game\_pause() { #暂定游戏

echo -en "\033[$Lines;$((Cols-50))H\033[33mGame paused, Use space or enter key to continue\033[0m";

while read -n 1 space; do

[[ ${space:-enter} = enter ]] && \

echo -en "\033[$Lines;$((Cols-50))H\033[33mPress <space> or enter to pause game \033[0m" && return;

[[ ${space:-enter} = q ]] && snake\_exit;

done

}

# $1 节点位置

update() { #更新各个节点坐标

case ${pos[$1]} in

right) ((ypt[$1]++));;

left) ((ypt[$1]--));;

down) ((xpt[$1]++));;

up) ((xpt[$1]--));;

esac

}

ch\_speed() { #更新速度，改变蛇的速度

[[ $# -eq 0 ]] && spk=$(((spk+1)%3));

case $spk in

0) temp="Fast ";;

1) temp="Mid";;

2) temp="Slow ";;

esac

echo -ne "\033[$Lines;3H\033[33mSpeed: $temp\033[0m";

}

Gooooo() { #更新方向，控制移动

case ${key:-enter} in

j|J) [[ ${pos[0]} != "up" ]] && pos[0]="down";;

u|U) [[ ${pos[0]} != "down" ]] && pos[0]="up";;

h|H) [[ ${pos[0]} != "right" ]] && pos[0]="left";;

k|K) [[ ${pos[0]} != "left" ]] && pos[0]="right";;

s|S) ch\_speed;;

q|Q) snake\_exit;;

enter) game\_pause;;

esac

}

add\_node() { #增加节点

snake="0$snake";

pos=(${pos[0]} ${pos[@]});

xpt=(${xpt[0]} ${xpt[@]});

ypt=(${ypt[0]} ${ypt[@]});

update 0;

local x=${xpt[0]} y=${ypt[0]}

(( ((x>=$((Lines-1)))) || ((x<=1)) || ((y>=Cols)) || ((y<=1)) )) && return 1; #撞墙

for (( i = $((${#snake}-1)); i > 0; i-- )); do

(( ${xpt[0]} == ${xpt[$i]} && ${ypt[0]} == ${ypt[$i]} )) && return 1; #crashed

done

echo -ne "\033[${xpt[0]};${ypt[0]}H\033[32m${snake[@]:0:1}\033[0m";

return 0;

}

mk\_random() { #产生随机点和随机数

xrand=$((RANDOM%(Lines-3)+2));

yrand=$((RANDOM%(Cols-2)+2));

foodscore=$((RANDOM%9+1));

echo -ne "\033[$xrand;${yrand}H$foodscore";

liveflag=0;

}

new\_game() { #重新开始新游戏

snake\_init;

while true; do

read -t ${speed[$spk]} -n 1 key;

[[ $? -eq 0 ]] && Gooooo;

((liveflag==0)) || mk\_random;

if (( sumnode > 0 )); then

((sumnode--));

add\_node; (($?==0)) || return 1;

else

update 0;

echo -ne "\033[${xpt[0]};${ypt[0]}H\033[32m${snake[@]:0:1}\033[0m";

for (( i = $((${#snake}-1)); i > 0; i-- )); do

update $i;

echo -ne "\033[${xpt[$i]};${ypt[$i]}H\033[32m${snake[@]:$i:1}\033[0m";

(( ${xpt[0]} == ${xpt[$i]} && ${ypt[0]} == ${ypt[$i]} )) && return 1; #crashed

[[ ${pos[$((i-1))]} = ${pos[$i]} ]] || pos[$i]=${pos[$((i-1))]};

done

fi

local x=${xpt[0]} y=${ypt[0]}

(( ((x>=$((Lines-1)))) || ((x<=1)) || ((y>=Cols)) || ((y<=1)) )) && return 1; #撞墙，超过所定义的边框范围。

(( x==xrand && y==yrand )) && ((liveflag=1)) && ((sumnode+=foodscore)) && ((sumscore+=foodscore));

echo -ne "\033[$xscore;$((yscore-2))H$sumscore";

done

}

----游戏结束

print\_good\_game() {

local x=$((xcent-4)) y=$((ycent-25))

for (( i = 0; i < 8; i++ )); do

echo -ne "\033[$((x+i));${y}H\033[45m${good\_game[$i]}\033[0m";

done

echo -ne "\033[$((x+3));$((ycent+1))H\033[45m${sumscore}\033[0m";

}

-----开始游戏

print\_game\_start() {

snake\_init;

local x=$((xcent-5)) y=$((ycent-25))

for (( i = 0; i < 10; i++ )); do

echo -ne "\033[$((x+i));${y}H\033[45m${game\_start[$i]}\033[0m";

done

while read -n 1 anykey; do

[[ ${anykey:-enter} = enter ]] && break;

[[ ${anykey:-enter} = q ]] && snake\_exit;

[[ ${anykey:-enter} = s ]] && ch\_speed;

done

while true; do

new\_game;

print\_good\_game;

while read -n 1 anykey; do

[[ $anykey = n ]] && break;

[[ $anykey = q ]] && snake\_exit;

done

done

}

----主函数

game\_main() {

trap 'snake\_exit;' SIGTERM SIGINT;

stty -echo; #取消回显

tput civis; #隐藏光标

tput smcup; clear; #保存屏幕并清屏

print\_game\_start; #开始游戏

}

game\_main;

五、实验心得

本次实验中我们做的是Linux环境下的贪吃蛇，首先在需要注意的是shell脚本的格式，要是 #!/ bin/ bash开头，之后再编译程序的时候需要注意的是程序是否具有执行权限，可以使用chmod来进行改写。在shell的相关函数的时候所需要的内容和C程序的使用有一定的差别，其中的编写规则有一些现在还有些不太理解，需要多次使用应能够解决。但有一些也是相差不大，如其中的背景色，只是使用具体号数即可。本次实验应操作次数有限，有许多内容还未了解透彻，应还需在实验外多次练习，熟练使用。