

LAPORAN PROYEK

Pemograman Sistem

SNAKE

Disusun untuk menyelesaikan Tugas Proyek Pemograman Sistem

Mahasiswa Institut Teknologi DEL



OLEH:

(13321020)	Kezia Claudia Simanjuntak
(13321022)	Haratama Felix Tamba
(13321030)	Erik Dito Tampubolon
(13321060)	Merry Margaretha Wijaya Tamba

INSTITUT TEKNOLOGI DEL
SITOLUAMA, LAGUBOTI
KABUPATEN TOBA SAMOSIR
TA 2022/2023

SNAKE

1. Deskripsi

Pada game ini sebagai pemain, kita akan mengendalikan sebuah makhluk yang menyerupai ular makanya game ini disebut snake yang dalam bahasa Indonesia artinya ular. Ular ini akan bergerak mengitari sebuah bidang berbentuk kotak, dengan tujuan mengambil makanan yang aslinya berbentuk titik atau kotak. Selama bermain, si pemain harus berusaha untuk tidak menabrak dinding atau ekornya sendiri dan itu akan semakin susah, karena setiap kali si pemain memakan makanan, ekornya akan bertambah panjang.

Di game ini implementasi IPC yang kami gunakan adalah “signal”, signal adalah pesan standar yang dikirim ke program yang sedang berjalan untuk memicu sebuah kejadian tertentu, seperti membuat berhenti nya sebuah program atau menangani kesalahan sebuah program. Signal ini biasanya digunakan di Sistem Operasi (OS) UNIX.

2. Tujuan

Game ini ditujukan untuk berbagai alasan seperti refreshing maupun bersenang – senang, game juga membawa dampak yang besar terutama pada perkembangan pola belajar dan prestasi seseorang serta pola berfikir seseorang dalam menghadapi sebuah persoalan. Maka dari itu kami membuat suatu game yang memberikan suatu hiburan dan pembelajaran tentang pola berfikir seseorang dalam menyelesaikan persoalan yang ada.

3. Cara bermain Snake

1. Pertama, ketika kita compile program menggunakan `chmod 755 snake.sh` lalu dilanjutkan `./snake.sh` maka akan muncul papan game snake yang sudah langsung berjalan.
2. Untuk mengatur snake mengambil makanan, maka kita menekan tombol
 - W = ke arah atas
 - A = ke arah kiri
 - S = ke arah tengah
 - D = ke arah kanan
3. Untuk keluar dari permainan kita dapat menekan tombol “q”.

4. Lakukan perintah sampai snake mengambil makanan dan akan terjadi perubahan pada tubuh snake yang akan semakin memanjang.
5. Jika snake sudah mendapatkan makanan dan tubuh memanjang maka score akan bertambah menjadi 1 dan seterusnya. Pertambahan akan terjadi setiap snake mendapatkan makanan.

6. Kode

```
GNU nano 2.3.1      File: snake.sh

#!/bin/bash

IFS=' '

declare -i height=$((($(tput lines)-5)) width=$((($(tput cols)-2))

# Pendeklarasian baris dan kolom pada program.
declare -i head_r head_c tail_r tail_c

declare -i alive
declare -i length
declare body

declare -i direction delta_dir
declare -i score=0

border_color="\e[30;43m"
snake_color="\e[32;42m"
food_color="\e[34;44m"
text_color="\e[31;43m"
no_color="\e[0m"

# Pendeklarasian signal yang digunakan pada program.
SIG_UP=USR1
SIG_RIGHT=USR2
SIG_DOWN=URG
SIG_LEFT=IO
SIG_QUIT=WINCH
SIG_DEAD=HUP

# Susunan arah ular: 0=up, 1=right, 2=down, 3=left
move_r=([0]=-1 [1]=0 [2]=1 [3]=0)
```

```

move_c=[0]=0 [1]=1 [2]=0 [3]=-1)

init_game() {
    clear
    echo -ne "\e[?25l"
    stty -echo
    for ((i=0; i<height; i++)); do
        for ((j=0; j<width; j++)); do
            eval "arr\${i}\${j}="
        done
    done
}

move_and_draw() {
    echo -ne "\e[${1}H${3}"
}

# Mencetak semua nya yang terdapat di dalam buffer.
draw_board() {
    move_and_draw 1 1 "$border_color+$no_color"
    for ((i=2; i<=width+1; i++)); do
        move_and_draw 1 $i "$border_color-$no_color"
    done
    move_and_draw 1 $((width + 2)) "$border_color+$no_color"
    echo

    for ((i=0; i<height; i++)); do
        move_and_draw $((i+2)) 1 "$border_color!$no_color"
        eval echo -en "\n\${arr\${i}*}\n"
        echo -e "$border_color!$no_color"
    done
}

```

```

move_and_draw $((height+2)) 1 "$border_color+$no_color"
for ((i=2; i<=width+1; i++)); do
    move_and_draw $((height+2)) $i "$border_color-$no_color"
done
move_and_draw $((height+2)) $((width + 2)) "$border_color+$no_color"
echo
}

# Mengatur profil snake
init_snake() {
    alive=0
    length=10
    direction=0
    delta_dir=-1

    head_r=$((height/2-2))
    head_c=$((width/2))

    body=''
    for ((i=0; i<length-1; i++)); do
        body="1$body"
    done

    local p=$((move_r[1] * (length-1)))
    local q=$((move_c[1] * (length-1)))

    tail_r=$((head_r+p))
    tail_c=$((head_c+q))

    eval "arr\$head_r[\$head_c]=\${snake_color}o\$no_color\""

    prev_r=$head_r

    prev_c=$head_c
    b=$body
    while [ -n "$b" ]; do
        # change in each direction
        local p=$((move_r[echo $b | grep -o '^[0-3]')])
        local q=$((move_c[echo $b | grep -o '^[0-3]')])

        new_r=$((prev_r+p))
        new_c=$((prev_c+q))

        eval "arr\$new_r[\$new_c]=\${snake_color}o\$no_color\""

        prev_r=$new_r
        prev_c=$new_c

        b=${b:1}
    done
}

is_dead() {
    if [ "$1" -lt 0 ] || [ "$1" -ge "$height" ] || \
        [ "$2" -lt 0 ] || [ "$2" -ge "$width" ]; then
        return 0
    fi
    eval "local pos=\${arr\$1[\$2]}"

    if [ "$pos" == "${snake_color}o\$no_color" ]; then
        return 0
    fi

    return 1
}

```

```

give_food() {
    local food_r=$((RANDOM % height))
    local food_c=$((RANDOM % width))
    eval "local pos=\${arr\$food_r[\$food_c]}"

    while [ "$pos" != ' ' ]; do
        food_r=$((RANDOM % height))
        food_c=$((RANDOM % width))
        eval "pos=\${arr\$food_r[\$food_c]}"
    done

    eval "arr\$food_r[\$food_c]=\${food_color}${no_color}"
}

move_snake() {
    local newhead_r=$((head_r + move_r[direction]))
    local newhead_c=$((head_c + move_c[direction]))

    eval "local pos=\${arr\$newhead_r[\$newhead_c]}"

    if $(is_dead $newhead_r $newhead_c); then
        alive=1
        return
    fi

    if [ "$pos" == "${food_color}${no_color}" ]; then
        length+=1
        eval "arr\$newhead_r[\$newhead_c]=\${snake_color}${no_color}"
        body="\${((direction+2)%4)}$body"
        head_r=$newhead_r
        head_c=$newhead_c

        score+=1
    fi

    give_food;
    return
fi
head_r=$newhead_r
head_c=$newhead_c

local d=$(echo $body | grep -o '[0-3]$')
body="\${((direction+2)%4)}${body%[0-3]}"

eval "arr\$tail_r[\$tail_c]=' '"
eval "arr\$head_r[\$head_c]=\${snake_color}${no_color}"

# Untuk mengatur pergerakan ekor ular
local p=$((move_r[(d+2)%4]))
local q=$((move_c[(d+2)%4]))
tail_r=$((tail_r+p))
tail_c=$((tail_c+q))
}

change_dir() {
    if [ $((direction+2)%4) -ne $1 ]; then
        direction=$1
    fi
    delta_dir=-1
}

getchar() {
    trap "" SIGINT SIGQUIT
    trap "return;" SIG_DIE

    while true; do

```

```

        read -s -n 1 key
        case "$key" in
            [qQ]) kill -SIG_QUIT $game_pid
                    return
                    ;;
            [wW]) kill -SIG_UP $game_pid
                    ;;
            [dD]) kill -SIG_RIGHT $game_pid
                    ;;
            [sS]) kill -SIG_DOWN $game_pid
                    ;;
            [aA]) kill -SIG_LEFT $game_pid
                    ;;
        esac
    done
}

game_loop() {
    trap "delta_dir=0;" SIG_UP
    trap "delta_dir=1;" SIG_RIGHT
    trap "delta_dir=2;" SIG_DOWN
    trap "delta_dir=3;" SIG_LEFT
    trap "exit 1;" SIG_QUIT

    while [ "$alive" -eq 0 ]; do
        echo -e "\n${text_color}Your score: $score $no_color"

        if [ "$delta_dir" -ne -1 ]; then
            change_dir $delta_dir
        fi
        move_snake
        draw_board

        sleep 0.03
    done

    echo -e "${text_color}Oh No! You dead please try again."

    # Memberi sinyal bahwa ular sudah mati.
    kill -SIG_DEAD $$
}

clear_game() {
    stty echo
    echo -e "\e[?25h"
}

init_game
init_snake
give_food
draw_board

game_loop &
game_pid=$?
getchar

clear_game
exit 0

```

```

^G Get Help      ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify      ^W Where Is     ^U Next Page    ^U UnCut Text   ^T To Spell

```

Cara Compile kode

```
[root@localhost proyek_081# chmod 755 snake.sh  
[root@localhost proyek_081# ./snake.sh_
```

Hasil Kode



snake.sh

```
#!/bin/bash
```

```
IFS=" # variabel ini menunjukkan bagaimana kata-kata dipisahkan pada baris perintah
```

```
declare -i height=$((($tput lines)-5)) width=$((($tput cols)-2)) # mendeklarasikan  
ketinggian dan tebal garis pada baris dan kolom
```

```
# Pendeklarasian baris dan kolom pada program
```

```
declare -i head_r head_c tail_r tail_c # digunakan untuk mendeklarasikan variabel dan  
fungsi di dalam program
```



```

declare -i alive # mendeklarasikan variabel ular hidup
declare -i length # mendeklarasikan variabel panjang ular
declare body # mendeklarasikan variabel body ular

declare -i direction delta_dir # mendeklarasikan arah ular
declare -i score=0 # mendeklarasikan score yang dimulai dari 0

border_color="\e[30;43m" # menentukan warna border
snake_color="\e[32;42m" # menentukan warna snake
food_color="\e[34;44m" # menentukan warna makanan
text_color="\e[31;43m" # menentukan warna teks
no_color="\e[0m" # menentukan bidang yang tidak terdapat warna

# Pendeklarasian signal yang digunakan pada program.
SIG_UP=USR1 # signal untuk memicu ular bergerak ke atas
SIG_RIGHT=USR2 # signal untuk memicu ular bergerak ke kanan
SIG_DOWN=URG # signal untuk memicu ular bergerak ke bawah
SIG_LEFT=IO # signal untuk memicu ular bergerak ke kiri
SIG_QUIT=WINCH # signal untuk menutup permainan
SIG_DEAD=HUP # signal untuk menandakan bahwa ular telah mati

# Susunan arah ular: 0=up, 1=right, 2=down, 3=left
move_r=([0]=-1 [1]=0 [2]=1 [3]=0)
move_c=([0]=0 [1]=1 [2]=0 [3]=-1)

# Menginisialisasi untuk membuat process baru
init_game() {
    clear
    echo -ne "\e[?25l"
    stty -echo
    for ((i=0; i<height; i++)); do
        for ((j=0; j<width; j++)); do
            eval "arr$i[$j]=' '"
        done
    done
}

# Nama fungsi untuk menggerakkan
move_and_draw() {
    echo -ne "\e[${1}];${2}H${3}"
}

# Mencetak papan bermain dan mewarnai garis tepi nya
draw_board() {
    move_and_draw 1 1 "$border_color+$no_color"

```

```

for ((i=2; i<=width+1; i++)); do
    move_and_draw 1 $i "$border_color-$no_color"
done
move_and_draw 1 $((width + 2)) "$border_color+$no_color"
echo

for ((i=0; i<height; i++)); do
    move_and_draw $((i+2)) 1 "$border_color|$no_color"
    eval echo -en "\"\${arr$i[*]}\\""
    echo -e "$border_color|$no_color"
done

move_and_draw $((height+2)) 1 "$border_color+$no_color"
for ((i=2; i<=width+1; i++)); do
    move_and_draw $((height+2)) $i "$border_color-$no_color"
done
move_and_draw $((height+2)) $((width + 2)) "$border_color+$no_color"
echo
}

# Menginisialisasi profil badan ular
init_snake() {
    alive=0
    length=10
    direction=0
    delta_dir=-1

    head_r=$((height/2-2))
    head_c=$((width/2))

    body=""
    for ((i=0; i<length-1; i++)); do
        body="$i$body"
    done
    local p=$(( ${move_r[1]} * (length-1) ))
    local q=$(( ${move_c[1]} * (length-1) ))

    tail_r=$((head_r+p))
    tail_c=$((head_c+q))

    eval "arr$head_r[$head_c]=\"${snake_color}o$no_color\""

    prev_r=$head_r
    prev_c=$head_c
    b=$body
    while [ -n "$b" ]; do

```

```

# Mengatur perubahan gerakan ular
local p=${move_r[(echo $b | grep -o '[0-3]')]}
local q=${move_c[(echo $b | grep -o '[0-3]')]}

new_r=$((prev_r+p))
new_c=$((prev_c+q))

eval "arr$new_r[$new_c]=\${snake_color}o$no_color\""

prev_r=$new_r
prev_c=$new_c

b=${b#[0-3]}
done
}

# Mengatur jika kondisi ular telah mati
is_dead() {
    if [ "$1" -lt 0 ] || [ "$1" -ge "$height" ] || \
        [ "$2" -lt 0 ] || [ "$2" -ge "$width" ]; then
        return 0
    fi
    eval "local pos=\${arr$1[$2]}"

    if [ "$pos" == "${snake_color}o$no_color" ]; then
        return 0
    fi

    return 1
}

# Mengatur posisi makanan secara acak
give_food() {
    local food_r=$((RANDOM % height))
    local food_c=$((RANDOM % width))
    eval "local pos=\${arr$food_r[$food_c]}"

    while [ "$pos" != '' ]; do
        food_r=$((RANDOM % height))
        food_c=$((RANDOM % width))
        eval "pos=\${arr$food_r[$food_c]}"
    done

    eval "arr$food_r[$food_c]=\${food_color@$no_color}"
}

```

Mengatur pergerakan ular

```
move_snake() {
    local newhead_r=$((head_r + move_r[direction]))
    local newhead_c=$((head_c + move_c[direction]))

    eval "local pos=\${arr$newhead_r[$newhead_c]}"

    if $(is_dead $newhead_r $newhead_c); then
        alive=1
        return
    fi

    if [ "$pos" == "$food_color@$no_color" ]; then
        length+=1
        eval "arr$newhead_r[$newhead_c]=\${snake_color}o$no_color\""
        body="$((((direction+2)%4))$body"
        head_r=$newhead_r
        head_c=$newhead_c

        score+=1
        give_food;
        return
    fi
    head_r=$newhead_r
    head_c=$newhead_c

    local d=$(echo $body | grep -o '[0-3]$')

    body="$((((direction+2)%4))${body%[0-3]})"

    eval "arr$tail_r[$tail_c]=''"
    eval "arr$head_r[$head_c]=\${snake_color}o$no_color\""

    # Untuk mengatur pergerakan ekor ular
    local p=${move_r[(d+2)%4]}
    local q=${move_c[(d+2)%4]}
    tail_r=$((tail_r+p))
    tail_c=$((tail_c+q))
}
```

Mengatur perubahan gerakan ular

```
change_dir() {
    if [ $((((direction+2)%4)) -ne $1 ); then
        direction=$1
    fi
    delta_dir=-1
}
```

```

}

# Membaca input-an yang dimasukkan oleh user, dan mengeksekusi fungsi nya
getchar() {
    trap "" SIGINT SIGQUIT
    trap "return;" SIG_DEAD

    while true; do
        read -s -n 1 key
        case "$key" in
            [qQ]) kill -$SIG_QUIT $game_pid
                    return
                    ;;
            [wW]) kill -$SIG_UP $game_pid
                    ;;
            [dD]) kill -$SIG_RIGHT $game_pid
                    ;;
            [sS]) kill -$SIG_DOWN $game_pid
                    ;;
            [aA]) kill -$SIG_LEFT $game_pid
                    ;;
            *)
                ;;
        esac
    done
}

# Mengeluarkan output jika kita telah kalah dalam permainan
game_loop() {
    trap "delta_dir=0;" SIG_UP
    trap "delta_dir=1;" SIG_RIGHT
    trap "delta_dir=2;" SIG_DOWN
    trap "delta_dir=3;" SIG_LEFT
    trap "exit 1;" SIG_QUIT

    while [ "$alive" -eq 0 ]; do
        echo -e "\n${text_color} Your score: $score $no_color"

        if [ "$delta_dir" -ne -1 ]; then
            change_dir $delta_dir
        fi
        move_snake
        draw_board
        sleep 0.03
    done

    echo -e "${text_color} Oh No! You dead please try again."
}

```

```

# Memberi sinyal bahwa ular sudah mati.
kill -SIG_DEAD $$
}
# Mengatur ulang papan bermain ketika kita memulai kembali permainan
clear_game() {
    stty echo
    echo -e "\e[?25h"
}

# Nama-nama fungsi yang di gunakan pada program
init_game
init_snake
give_food
draw_board

game_loop &
game_pid=$!
getchar

# Fungsi untuk memberikan perintah bawasannya game sudah berakhir dan user bisa
# memulai game kembali
clear_game
exit 0

```

Penjelasan program:

Baris 1 : /bin/bash adalah shell yang paling umum digunakan sebagai shell default untuk login pengguna sistem linux.

Baris 3: variabel ini menunjukkan bagaimana kata-kata dipisahkan pada baris perintah. Variabel IFS, biasanya atau secara default, spasi putih (' '). Variabel IFS digunakan sebagai pemisah kata (token) untuk perintah for.

Baris 5 : mendeklarasikan ketinggian dan tebal garis pada baris dan kolom.

Baris 8 : digunakan untuk mendeklarasikan variabel dan fungsi di dalam program.

Baris 11 : mendeklarasikan variabel ular hidup, jadi kita harus membuat sebuah fungsi yang dimana fungsi ini akan menyatakan bahwa ular ini hidup.

Baris 12 : mendeklarasikan variabel panjang ular, kita membuat fungsi yang berguna untuk menyatakan panjang ular tersebut.

Baris 13 : mendeklarasikan variabel body ular, kita membuat fungsi yang berguna menyatakan badan ular akan seperti apa ketika di dalam permainan tersebut. Badan ular ketika di dalam permainan bisa menjadi melengkung ketika melakukan belok atau ular bisa mati ketika kepala ular menyentuh badannya sendiri.

Baris 15 : mendeklarasikan arah ular, kita membuat fungsi yang berguna menyatakan arah gerakan ular. Badan ular ketika di dalam permainan bisa belok ke kanan atau kiri, dan bisa juga ke arah atas ataupun bawah.

Baris 16 : mendeklarasikan score yang dimulai dari 0, kita membuat fungsi yang berguna menampilkan score awal ketika permainan dimulai, jadi secara default ketika kita memulai permainan ini score awal nya adalah 0, kemudian score ini akan bertambah seiring ular tersebut memakan makanannya.

Baris 18 : menentukan warna border permainan.

Baris 19 : menentukan warna snake.

Baris 20 : menentukan warna makanan.

Baris 21 : menentukan warna teks.

Baris 22 : menentukan bidang yang tidak terdapat warna.

Baris 25 : memberikan sebuah signal yang dimana signal ini akan memicu sebuah kejadian (event) yang menggerakkan ular ke atas.

Baris 26 : memberikan sebuah signal yang dimana signal ini akan memicu sebuah kejadian (event) yang menggerakkan ular ke kanan.

Baris 27 : memberikan sebuah signal yang dimana signal ini akan memicu sebuah kejadian (event) yang menggerakkan ular ke bawah.

Baris 28 : memberikan sebuah signal yang dimana signal ini akan memicu sebuah kejadian (event) yang menggerakkan ular ke kiri.

Baris 29 : memberikan sebuah signal yang dimana signal ini akan memicu sebuah kejadian (event) yang akan menutup permainan.

Baris 30 : memberikan sebuah signal bahwa ular telah mati.

Baris 33-34 : ini adalah susunan arah ular, angka 0 untuk ke atas, angka 1 untuk ke kanan, angka 2 untuk ke bawah dan angka 3 untuk ke kiri.

Baris 37-46 : menginisialisasi untuk membuat process baru

Baris 49-51 : fungsi untuk mengatur papan (board) permainan ketika ular bergerak.

Baris 54-74 : mencetak papan bermain dan mewarnai garis tepi ketika permainan dimulai.

Baris 77-116 : mendeklarasikan kondisi badan ular ketika permainan baru dimulai, ketika permainan dimulai panjang ular sebanyak 10 balok dan ular tersebut berjalan secara statis sampai pengguna melakukan kontrol terhadap ular dengan cara menekan tombol WASD untuk mengontrol arah ular.

Baris 118-131 : mengatur jika kondisi ular telah mati.

Baris 134-146 : mengatur posisi makanan secara acak, ketika ular berhasil memakan makanan maka panjang ular akan bertambah dan makanan akan muncul lagi di posisi yang acak.

Baris 149-179 : mengatur pergerakan ular, ular akan bergerak sesuai dengan input-an yang diberikan oleh pengguna, jika pengguna menekan tombol “W” maka ular akan bergerak ke atas, jika menekan tombol “A” maka ular akan bergerak ke kiri, jika menekan tombol “S” ular akan bergerak ke bawah dan jika menekan tombol “D” ular akan bergerak ke kanan.

Baris 182- 186 : mengatur pergerakan ekor ular, jika ular berbelok ke arah yang lain ekor ular akan mengikuti pergerakan kepala ular.

Baris 189-194 : mengatur perubahan gerakan ular ketika pengguna mengontrol ular melalui keyboard.

Baris 197-217 : membaca input-an yang dimasukkan oleh user, dan mengeksekusi fungsi nya. Jadi ketika pengguna memasukkan sebuah input-an, contohnya menekan tombol “A”, maka ular akan bergerak ke kiri dan jika pengguna menekan tombol “Q”, maka permainan akan selesai.

Baris 220-238 : mengeluarkan output jika kita telah kalah dalam permainan, jika kita telah kalah dalam permainan (ularnya mati) maka program akan mengeluarkan output score yang sudah kita peroleh dan menampilkan teks “Oh No! You dead please try again.”.

Baris 241 : memberi signal bahwa ular sudah mati.

Baris 244-247 : mengatur ulang papan bermain ketika kita memulai kembali permainan

Baris 250-257 : nama-nama fungsi yang di gunakan pada program

Baris 260-261 : fungsi untuk memberikan perintah bawasannya game sudah berakhir dan user bisa memulai game kembali