Universidade de Brasília
Engenharia Mecânica
Carlos Adir Ely Murussi Leite

Algoritmos e Programação de Computadores

Guia ao *kbhit* e *getch*

A utilidade

Enquanto estamos desenvolvendo algo, frequentemente em um jogo ou fazer um login de um usuário no seu programa, queremos que o programa funcione enquanto o usuário não faz nada ou mesmo que capture a entrada do usuário sem apertar enter. Para isso que servem as duas funções:

- *kbhit* - Essa função indica se existe uma entrada no buffer
- *getch* - Essa função captura um caracter do buffer sem que seja necessário apertar enter

Login de usuário

Quando se faz o login do usuário, é interessante esconder a senha à medida que o usuário digite. Os programas feitos até aqui sempre que você digita qualquer caracter, ele fica exposto até você apertar enter(enviar para processamento). Logo, queremos que quando o usuário digite um **char**, o programa capture esse caracter e escreva o **'*'** no seu lugar.

Jogo 2048

No Jogo 2048, quando se toma uma decisão(enviar as peças para algum lugar), se for necessário apertar **enter** para cada comando, o jogo fica extremamente ruim.

Jogo Snake

No Jogo Snake, enquanto o usuário não digita algo, a serpente se move até o usuário decidir fazer algo.

Os requisitos

Ambas as funções foram testadas utilizando o compilador gcc versão 5.4.0 utilizando Ubuntu 16.04.4. Versões similares do gcc e na maioria das distribuições linux são capazes de compilar e executar, além de ter as bibliotecas(headers) requeridas.

getch

Para a função *getch* é necessário três *headers*:
1) stdio
2) termios
3) unistd

kbhit

Para a função *kbhit* é necessário quatro *headers*:
1) stdio
2) termios
3) unistd
4) fcntl

Obs: Não foram testadas estas funções para windows.

O código

Por questões de complexidade de código, o código não será explicado. Mas como não são funções padrões então é necessário que tenhamos o código. O código é dado tanto pelo arquivo kbhitgetch.h quanto é mostrado abaixo.

getch

```
char getch()
{
    int ch;
    struct termios oldt;
    struct termios newt;
    tcgetattr(STDIN_FILENO,&oldt);
    newt = oldt;
    newt.c_lflag &= ~(ICANON | ECHO);
    tcsetattr(STDIN_FILENO, TCSANOW, &newt);
    ch = getchar();
    tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
    return ch;
}
```

kbhit

```
int kbhit(void)
{
  struct termios oldt, newt;
  int ch;
  int oldf;

  tcgetattr(STDIN_FILENO, &oldt);
  newt = oldt;
  newt.c_lflag &= ~(ICANON | ECHO);
  tcsetattr(STDIN_FILENO, TCSANOW, &newt);
  oldf = fcntl(STDIN_FILENO, F_GETFL, 0);
  fcntl(STDIN_FILENO, F_SETFL, oldf | O_NONBLOCK);

  ch = getchar();

  tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
  fcntl(STDIN_FILENO, F_SETFL, oldf);

  if(ch != EOF)
  {
    ungetc(ch, stdin);
    return 1;
  }

  return 0;
}
```

O funcionamento

Agora explicaremos como funcionam as funções *getch* e *kbhit*

getch

Essa função é semelhante ao getchar. A diferença de uso de ambas as funções é que a *getch* não precisa esperar o usuário terminar de digitar e depois apertar enter.

Um exemplo de implementação é mostrado pelo arquivo 1-getch_example1.c, em que cada letra que é digitada é mostrada na tela até que aperte enter('**n**').

```
1  #include <stdio.h>
2  #include "kbhitgetch.h"
3
4  int main()
5  {
6    char c;
7    do{
8      c = getch();
9      putchar(c);
10   }while(c != '\n');
11   return 0;
12 }
```

Bem, se você for bastante esperto, notará a diferença entre antes e depois. Caso ainda não tenha notado, primeiro digite alguns caracteres(no terminal) e tente apagá-los. Caso ainda não tenha notado a diferença, apague a linha 9 do código acima e teste.

Outra diferença que pode-se notar é que quando você digita um caracter quando há o *getch* é que ele não parece na tela. No exemplo acima, só apareceram os caracteres devido à linha 9 que contém um *putchar*.

Agora, outro exemplo(2-getch_example2.c) para o caso do **login** que foi mostrado no inicio, é trocar na linha 9 o argumento de **c** para **'*'**. E então mostrará escondida a senha e informará ao usuário quantos caracteres. Mais abaixo iremos retomar este exemplo e como podemos melhorar.

kbhit

Essa função verifica no *buffer* se existe alguma entrada. Caso haja alguma entrada, então a função retorna 1 e caso contrário retorna 0.

Um exemplo de implementação é mostrado pelo arquivo 3-kbhit_example1.c, em que imprime a mensagem "**Aperte uma tecla para parar**" até que se aperte uma tecla.

```
1  #include <stdio.h>
2  #include "kbhitgetch.h"
3
4  int main()
5  {
6    while(!kbhit()) /* Enquanto nao tem entrada */
7    {
8      printf("Aperte uma tecla para parar\n");
9    }
10   printf("Voce parou o processo apertando alguma tecla!\n\n");
11   return 0;
12 }
```

O importante é que essa função não captura o caracter digitado. Ela simplesmente verifica se tem entrada ou não. Essa entrada, assim como no *getch* reconhece antes de apertar enter.

Neste caso, se tiver entrada, podemos capturar essa entrada utilizando uma entrada de dados que pode ser *getchar* ou a função que acabamos de aprender que é *getch*.

Assim, no código abaixo(4-kbhit_example2.c) está um uso das duas funções juntas que semelhantemente ao código acima, pede para parar o processo apertando uma tecla. A diferença é que agora iremos informar qual foi o caracter digitado.

```c
#include <stdio.h>
#include "kbhitgetch.h"

int main()
{
  while(!kbhit()) /* Enquanto nao tem entrada */
  {
    printf("Aperte uma tecla para parar\n");
  }
  printf("Voce parou o processo apertando a tecla: [%c]!\n\n", getch
     ());
  return 0;
}
```

Agora, faremos um número começar do zero e para cada aperto em uma tecla, informará um número correspondente. O código é 5-kbhit_example3.c

```c
#include <stdio.h>
#include "kbhitgetch.h"

int main()
{
  long long int contador = 0;
  char c = 'a';
  while(c != '\n')/* Enquanto o digitado nao for digitado o Enter */
  {
    contador += 1;
    if(kbhit()) /* Se existe entrada */
    {
      c = getch(); /* Entao captura e grava na variavel c */
      printf("contador = %d\n", contador); /* E imprime o numero
         correspondente */
    }
  }
  return 0;
}
```

Neste código, quanto maior o intervalo entre uma tecla e outra, mais distante um número estará do outro.

Aplicações com getch e kbhit

Como de costume, sempre que aprendemos algo novo queremos implementar. Você tem capacidade de implementar algumas coisas interessantes já, como por exemplo o Jogo Snake. Mas tem algumas coisas interessantes que ainda não vimos...

Teclas especiais

Como que captura uma seta para cima ou a tecla F2?

Vá no código 2-getch_example2.c e digite a tecla BACKSPACE e veja o que acontece. Se você digitar, aparecerá um caracter que segundo o código getch_example1.c não existe. Bem, na verdade existe, a diferença é que quando você tenta imprimi-lo, imprime nada.

Agora, nesse mesmo código, digite a seta para cima (as direcionais) e verá que aparecerá três simbolos '*'. Ora, aprendemos algo inusitado, mas de princípio não sabemos o que é. Para auxiliar na nossa análise do que é, vamos em vez de imprimir '*', iremos imprimir o próprio caracter. Contudo, como no caso do BACKSPACE, imprimindo o próprio caracter não é muito inteligente e neste caso imprimeremos o número correspondente. Para imprimir o número faremos o uso da tabela ASCII. Alguns caracteres e seus respectivos números no padrão ASCII estão no quadro:

Dec	Char	Dec	Char	Dec	Char	Dec	Char
48	0	58	:	65	A	97	a
49	1	59	;	66	B	98	b
50	2	60	<	67	C	99	c
51	3	61	=	68	D	100	d
52	4	62	>	69	E	101	e
53	5	63	?	86	V	118	v
54	6	64	@	87	W	119	w
55	7	42	*	88	X	120	x
56	8	43	+	89	Y	121	y
57	9	45	-	90	Z	122	z

Assim, um código que lê um char e escreve o número correspondente é dado por 6-aplicacao.c que é mostrado abaixo:

```c
#include <stdio.h>
#include "kbhitgetch.h"

int main()
{
  char c;
  do{
    c = getch();
    printf("%d\n", c);
  }while(c != '\n');
  return 0;
}
```

Assim você consegue descobrir os números correspondente segundo a tabela ASCII. Agora, o que acontece se apertar a seta para cima? Irá imprimir três números consecutivos que são 27, 91 e 65. Assim que funcionam as teclas que não são "simples", elas são decompostas em uma sequência de caracteres que no caso da seta para cima são respectivamente 27, 91 e 65. Para cada tecla, é uma outra sequência correspondente. No caso do F2, é a sequência 27, 79 e 81. Para a tecla "Page Up" é a sequência 27, 91, 53 e 126.

Quando você tenta aperta uma determinada tecla, uma sequência de comandos irá para o buffer e não importa quão veloz você seja, nunca irá apertar duas teclas ao mesmo tempo e isso é uma vantagem para nós, pois iremos utilizar justamente isso para diferenciar as teclas 'ESC'(27), '^ '(94) e 'A'(65) da UP_KEY(27, 94, 65).

Para arrumar isso, vamos verificar se existe entrada após digitar uma tecla. Caso exista ele vai armazenar novamente em outra variável e verifica se existe novamente entrada e repete o processo até que acabe. Assim, se apertarmos a sequência 'ESC'+'^ '+'A' haverá uma pausa entre as teclas enquanto que digitar UP_KEY todas são enviadas de uma vez.

O código 7-get_keys.c exemplifica esse processo. Caso seja pegos todo de uma vez, então imprime todos em uma linha. Teste o código e veja como funciona. Para sair, basta apertar enter ou mesmo Ctrl+C.

```c
char c;
do{
  do{
    c = getch();
    printf("%d;", c);
  }while(kbhit()); /* Enquanto houver entrada, ler e imprimir */
  printf("\n"); /* Imprimir a quebra de linha para diferenciar
     comandos */
}while(c != '\n'); /* Pode haver confusao se o ultimo caracter da
   sequencia tambem tiver numero 65, mas muito raro */
return 0;
```

A partir daí, podemos fazer uma função que verifica isso e retorna um número correspondente à tecla, mas como é um conjunto de números(lembre-se, 27 94 65), então é necessário criar nosso próprio padrão. Um padrão bom para isso é mostrado no código funcoes_extras.c, em que se utiliza uma função que retorna inteiro e dependendo da entrada, consegue retornar uma tecla de até 8 chars.

```c
unsigned long int key()
{
  int j, i = 0;
  unsigned long int retorno = 0;
  char c[8];
  do{
    c[i] = getch();
    i++;
  }while(kbhit() && i < 8);
  for(j = 0; j < i; j++)
  {
    retorno *= 256;
    if(c[j] < 0)
      retorno += 256;
    retorno += c[j];
  }
  return retorno;
}
```

Assim, tem-se que para um número, sempre há somente uma combinação devido à base numérica utilizada. Tal função utiliza somente uma variável, contudo há possibilidade de retornar uma estrutura ou gravar os chars dentro de um vetor utilizando ponteiros.

Entendendo o funcionamento da função(*key*) pode-se criar diversas outras que sejam mais fáceis de utilizar. Algumas chaves estão armazenadas no arquivo e diversas outras podem ser descobertas fazendo combinações. Um exemplo de combinação é utilizar *Alt + alguma_tecla*, que no caso $Alt + q = 27\ 113$ enquanto $q = 113$.

Outras funções

Outras funções estão no arquivo funcoes_extras.c, documentadas e explicando o que cada uma faz além de serem fáceis de entender sabendo o que as funções *kbhit* e *getch* fazem.

Contato

Para críticas, sugestões, dúvidas e/ou contribuições, envie e-mail para:
carlos.adir.leite@gmail.com.