

# Princípios de Desenvolvimento de Algoritmos MAC122

Prof. Dr. Paulo Miranda  
**IME-USP**

Vazamento de memória &  
Erros em tempo de execução

# Vazamento de memória

- **Introdução:**

- Vazamento de memória (**memory leak**), é um problema que ocorre em sistemas computacionais quando uma porção de memória, alocada para uma determinada operação, não é liberada quando não é mais necessária.
- Vazamentos de memória estão entre os erros (**bugs**) mais difíceis de detectar, porque eles não causam nenhum problema aparente, até você ficar sem memória e sua chamada a **malloc** falhar de repente.

# Vazamento de memória

- **Exemplo:**

- A função aloca no heap um vetor de 100 bytes que não são desalocados.

```
#include <stdlib.h>
void teste(){
    char *p;
    p = (char *)malloc(100*sizeof(char));

    /* processa operação */
    ...
}
int main(){
    int i = 0;
    while (i < 100) {
        teste();
        i++;
    }
    return 0;
}
```

# Vazamento de memória

- **Problema:**

- No exemplo anterior o erro era evidente, porém muitas vezes a função é chamada poucas vezes e o problema passa despercebido sem maiores consequências.
- No entanto, a medida que os programas crescem esses erros se tornam cada vez mais difíceis de serem detectados, e o problema pode passar a ter proporções maiores levando a falhas no sistema se a memória for completamente consumida.

# Vazamento de memória

- **Solução:**

- A solução é utilizar um software depurador para auxiliar a detectar os vazamentos.
- “valgrind” é uma opção disponível no ambiente linux:
  - Ele possui ferramentas que detectam erros decorrentes do uso incorreto da memória dinâmica, como por exemplo:
    - os vazamentos de memória,
    - alocação e desalocação incorretas e
    - acessos a áreas inválidas.

# Vazamento de memória

- **valgrind:**

- Assumindo que o programa anterior está em um arquivo chamado “teste.c”, podemos depurar o vazamento de memória usando os comandos:

```
gcc teste.c -g -o teste
```

```
valgrind --tool=memcheck --leak-check=yes ./teste
```

# Vazamento de memória

- **valgrind:**

- Assumindo que o programa chamado “teste” não libera a memória usada.

Para produzir informações de depuração (ex: símbolos de depuração). Sem essa opção o valgrind não vai conseguir informar a linha do código onde está o problema.

```
gcc teste.c -g -o teste
```

```
valgrind --tool=memcheck --leak-check=yes ./teste
```

# Valgrind

- Exemplo de execução:

```
pavm@pavm-comp: ~/Desktop/registros/vazamento
==7918== Command: ./teste
==7918==
==7918==
==7918== HEAP SUMMARY:
==7918==   in use at exit: 10,000 bytes in 100 blocks
==7918== total heap usage: 100 allocs, 0 frees, 10,000 bytes allocated
==7918==
==7918== 10,000 bytes in 100 blocks are definitely lost in loss record 1 of 1
==7918==   at 0x4C28FAC: malloc (vg_replace_malloc.c:236)
==7918==   by 0x400505: teste (teste.c:6)
==7918==   by 0x400526: main (teste.c:14)
==7918==
==7918== LEAK SUMMARY:
==7918==   definitely lost: 10,000 bytes in 100 blocks
==7918==   indirectly lost: 0 bytes in 0 blocks
==7918==   possibly lost: 0 bytes in 0 blocks
==7918==   still reachable: 0 bytes in 0 blocks
==7918==   suppressed: 0 bytes in 0 blocks
==7918==
==7918== For counts of detected and suppressed errors, rerun with: -v
==7918== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 4 from 4)
pavm@pavm-comp:~/Desktop/registros/vazamento$
```



# Valgrind

- Exemplo de execução:

100 chamadas de **malloc**, e zero chamadas de **free**.

```
pavm@pavm-comp: ~/Desktop/registros/v
==7918== Command: ./teste
==7918==
==7918==
==7918== HEAP SUMMARY:
==7918==   in use at exit: 10,000 bytes in 100 blocks
==7918==   total heap usage: 100 allocs, 0 frees, 10,000 bytes allocated
==7918==
==7918== 10,000 bytes in 100 blocks are definitely lost in loss record 1 of 1
==7918==   at 0x4C28FAC: malloc (vg_replace_malloc.c:236)
==7918==   by 0x400505: teste (teste.c:6)
==7918==   by 0x400526: main (teste.c:14)
==7918==
==7918== LEAK SUMMARY:
==7918==   definitely lost: 10,000 bytes in 100 blocks
==7918==   indirectly lost: 0 bytes in 0 blocks
==7918==   possibly lost: 0 bytes in 0 blocks
==7918==   still reachable: 0 bytes in 0 blocks
==7918==   suppressed: 0 bytes in 0 blocks
==7918==
==7918== For counts of detected and suppressed errors, rerun with: -v
==7918== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 4 from 4)
pavm@pavm-comp:~/Desktop/registros/vazamento$
```

# Valgrind

- Exemplo de execução:

```
pavm@pavm-comp: ~/Desktop/registros/vazamento
==7918== Command: ./teste
==7918==
==7918== HEAP SUMMARY:
==7918==   in use at exit: 10,000 bytes in 100 blocks
==7918==   total heap usage: 100 allocs, 0 frees, 10,000 bytes allocated
==7918==
==7918== 10,000 bytes in 100 blocks are definitely lost
==7918==   at 0x4C28FAC: malloc (vg_replace_malloc.c:39)
==7918==   by 0x400505: teste (teste.c:6)
==7918==   by 0x400526: main (teste.c:14)
==7918==
==7918== LEAK SUMMARY:
==7918==   definitely lost: 10,000 bytes in 100 blocks
==7918==   indirectly lost: 0 bytes in 0 blocks
==7918==   possibly lost: 0 bytes in 0 blocks
==7918==   still reachable: 0 bytes in 0 blocks
==7918==   suppressed: 0 bytes in 0 blocks
==7918==
==7918== For counts of detected and suppressed errors, rerun with: -v
==7918== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 4 from 4)
pavm@pavm-comp:~/Desktop/registros/vazamento$
```

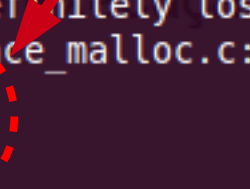
**Vazamento total:**  
100 execuções do laço, e cada chamada da função aloca 100 bytes:  
  
100x100 = 10.000 bytes

# Valgrind

- Exemplo de execução:

```
pavm@pavm-comp: ~/Desktop/registros/vazam
==7918== Command: ./teste
==7918==
==7918== HEAP SUMMARY:
==7918==   in use at exit: 10,000 bytes in 100 blocks
==7918==   total heap usage: 100 allocs, 0 frees, 10,000 bytes allocated
==7918==
==7918== 10,000 bytes in 100 blocks are definitely lost in 1 record 1 of 1
==7918==   at 0x4C28FAC: malloc (vg_replace_malloc.c:236)
==7918==   by 0x400505: teste (teste.c:6)
==7918==   by 0x400526: main (teste.c:14)
==7918==
==7918== LEAK SUMMARY:
==7918==   definitely lost: 10,000 bytes in 100 blocks
==7918==   indirectly lost: 0 bytes in 0 blocks
==7918==   possibly lost: 0 bytes in 0 blocks
==7918==   still reachable: 0 bytes in 0 blocks
==7918==   suppressed: 0 bytes in 0 blocks
==7918==
==7918== For counts of detected and suppressed errors, rerun with: -v
==7918== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 4 from 4)
pavm@pavm-comp:~/Desktop/registros/vazamento$
```

Ordem das funções na pilha e **linhas** onde aconteceram as chamadas.



# Acessos a áreas inválidas

- **Exemplo:**

- O vetor tem 100 posições, numeradas de 0 a 99.

```
#include <stdlib.h>

void teste(){
    char *p;
    p = (char *)malloc(100*sizeof(char));

    p[100] = 'A'; /* acesso inválido */

    free(p);
}

int main(){

    teste();
    return 0;
}
```

# Valgrind

- Exemplo de execução:

```
pavm@pavm-comp: ~/Desktop/registros/vazamento
==8194== Command: ./teste
==8194==
==8194== Invalid write of size 1
==8194==    at 0x400562: teste (teste.c:8)
==8194==    by 0x400580: main (teste.c:15)
==8194==   Address 0x51c30a4 is 0 bytes after a block of size 100 alloc'd
==8194==   at 0x4C28FAC: malloc (vg_replace_malloc.c:236)
==8194==   by 0x400555: teste (teste.c:6)
==8194==   by 0x400580: main (teste.c:15)
==8194==
==8194==
==8194== HEAP SUMMARY:
==8194==   in use at exit: 0 bytes in 0 blocks
==8194== total heap usage: 1 allocs, 1 frees, 100 bytes allocated
==8194==
==8194== All heap blocks were freed -- no leaks are possible
==8194==
==8194== For counts of detected and suppressed errors, rerun with: -v
==8194== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 4 from 4)
pavm@pavm-comp:~/Desktop/registros/vazamento$
```

# Valgrind

- Exemplo de execução:

```
pavm@pavm-comp: ~/Desktop/registros/vazament
==8194== Command: ./teste
==8194==
==8194== Invalid write of size 1
==8194==   at 0x400562: teste (teste.c:8)
==8194==   by 0x400580: main (teste.c:15)
==8194== Address 0x51c30a4 is 0 bytes after a block of size 100 alloc'd
==8194==   at 0x4C28FAC: malloc (vg_replace_malloc.c:236)
==8194==   by 0x400555: teste (teste.c:6)
==8194==   by 0x400580: main (teste.c:15)
==8194==
==8194==
==8194== HEAP SUMMARY:
==8194==   in use at exit: 0 bytes in 0 blocks
==8194== total heap usage: 1 allocs, 1 frees, 100 bytes allocated
==8194==
==8194== All heap blocks were freed -- no leaks are possible
==8194==
==8194== For counts of detected and suppressed errors, rerun with: -v
==8194== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 4 from 4)
pavm@pavm-comp:~/Desktop/registros/vazamento$
```

Acesso inválido

# Erros em tempo de execução

- **Exemplo:**

- O seguinte código apresenta um erro em tempo de execução, que não é detectado durante a compilação.

```
#include <stdlib.h>

void teste(int *p) {
    *p = 12;
}

int main() {

    teste(NULL);
    return 0;
}
```

# Erros em tempo de execução

- **Exemplo:**

- O seguinte código apresenta um erro em tempo de execução, que não é detectado durante a compilação.

```
#include <stdlib.h>

void teste(int *p) {
    *p = 12;
}

int main() {
    teste(NULL);
    return 0;
}
```

O problema acontece porque o apontador p recebe o valor **NULL**, o que significa que ele não está apontando para ninguém.



# Erros em tempo de execução

- **Exemplo:**

- O seguinte código apresenta um erro em tempo de execução, que não é detectado durante a compilação.

```
#include <stdlib.h>
```

```
void teste(int *p) {  
    *p = 12;  
}
```

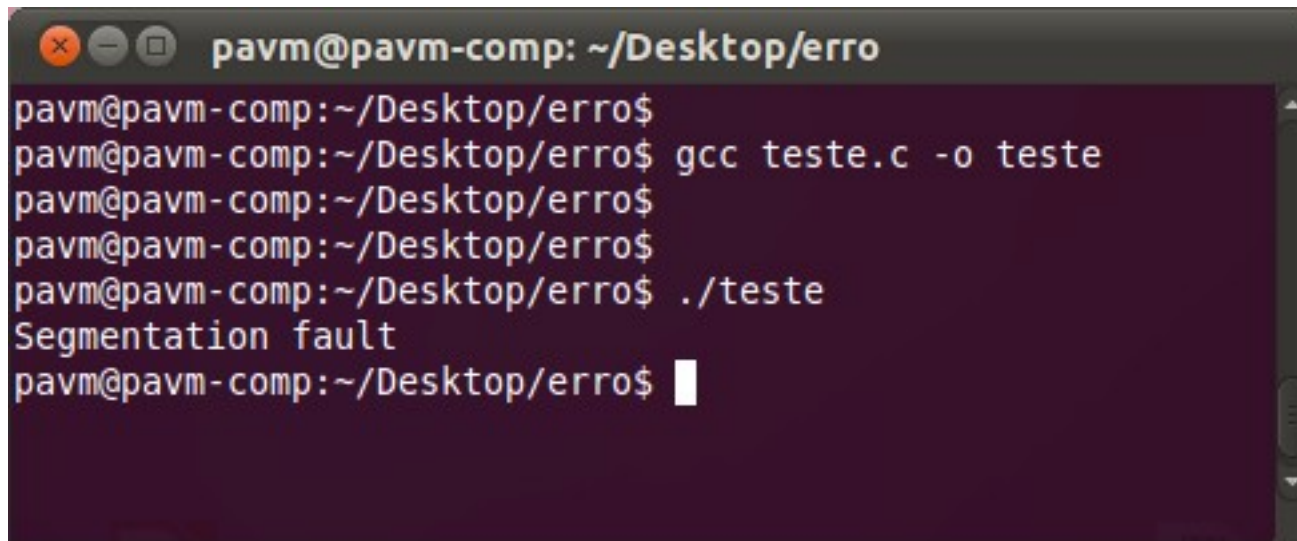
```
int main() {  
  
    teste(NULL);  
    return 0;  
}
```

O conteúdo da variável apontada \*p não existe, visto que p não aponta para ninguém.

# Erros em tempo de execução

- **Exemplo:**

- O seguinte código apresenta um erro em tempo de execução, que não é detectado durante a compilação.

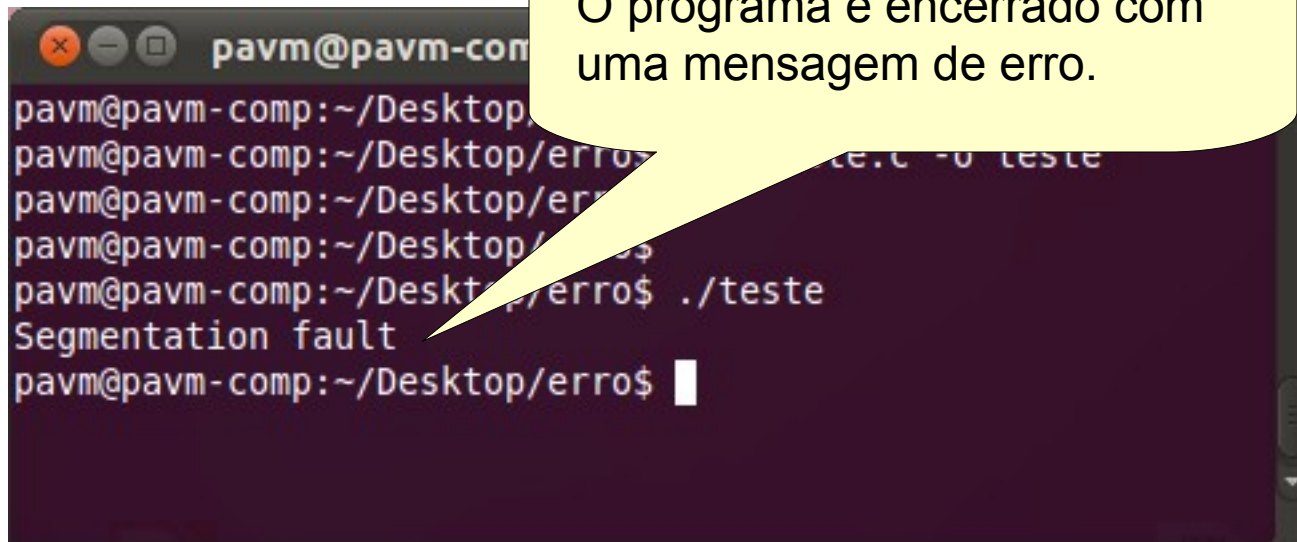


```
pavm@pavm-comp: ~/Desktop/erro
pavm@pavm-comp:~/Desktop/erro$
pavm@pavm-comp:~/Desktop/erro$ gcc teste.c -o teste
pavm@pavm-comp:~/Desktop/erro$
pavm@pavm-comp:~/Desktop/erro$ ./teste
Segmentation fault
pavm@pavm-comp:~/Desktop/erro$
```

# Erros em tempo de execução

- **Exemplo:**

- O seguinte código apresenta um erro em tempo de execução, que não é detectado durante a compilação.



A terminal window with a dark background and light text. The window title is 'pavm@pavm-comp'. The prompt is 'pavm@pavm-comp:~/Desktop/errores'. The user enters 'te.c -o teste'. The prompt changes to 'pavm@pavm-comp:~/Desktop/errores\$'. The user enters './teste'. The output is 'Segmentation fault'. The prompt returns to 'pavm@pavm-comp:~/Desktop/errores\$' with a cursor. A yellow speech bubble points to the 'Segmentation fault' message.

```
pavm@pavm-comp:~/Desktop/errores$ gcc te.c -o teste
pavm@pavm-comp:~/Desktop/errores$ ./teste
Segmentation fault
pavm@pavm-comp:~/Desktop/errores$
```

O programa é encerrado com uma mensagem de erro.

# Erros em tempo de execução

- **Solução geral:**

- No caso de um código mais complexo, esse tipo de erro pode ser mais facilmente detectado usando o depurador GDB.
- GDB, o depurador do projeto GNU, permite que você veja o que está acontecendo 'dentro' de outro programa enquanto ele executa - ou o que o outro programa estava fazendo no momento em que ele travou.
- Alguns comandos:

```
gcc teste.c -g -o teste
```

```
gdb ./teste
```

```
(gdb) run
```

```
(gdb) backtrace
```

```
(gdb) q
```

Para iniciar

Mostra as chamadas da pilha

Para encerrar (quit)

# Depurador GDB

```
pavm@pavm-comp: ~/Desktop/erro
pavm@pavm-comp:~/Desktop/erro$ gdb ./teste
GNU gdb (Ubuntu/Linaro 7.2-1ubuntu11) 7.2
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/pavm/Desktop/erro/teste...done.
(gdb) run
Starting program: /home/pavm/Desktop/erro/teste

Program received signal SIGSEGV, Segmentation fault.
0x00000000004004c0 in teste (p=0x0) at teste.c:5
5      *p = 12;
(gdb) backtrace
#0  0x00000000004004c0 in teste (p=0x0) at teste.c:5
#1  0x00000000004004d6 in main () at teste.c:10
(gdb) █
```

# Depurador GDB

```
pavm@pavm-comp: ~/Desktop/erro
pavm@pavm-comp:~/Desktop/erro$ gdb ./teste
GNU gdb (Ubuntu/Linaro 7.2-1ubuntu11) 7.2
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/pavm/Desktop/erro/teste
(gdb) run
Starting program: /home/pavm/Desktop/erro/teste
Program received signal SIGSEGV, Segmentation fault.
0x00000000004004c0 in teste (p=0x0) at teste.c:5
5      *p = 12;
(gdb) backtrace
#0  0x00000000004004c0 in teste (p=0x0) at teste.c:5
#1  0x00000000004004d6 in main () at teste.c:10
(gdb) █
```

Mostra a linha onde aconteceu o problema, e a instrução que foi responsável pelo erro.