Základy programování (IZP)

Osmé počítačové cvičení

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 66 Brno - Královo Pole
Petr Veigend, iveigend@fit.vut.cz, Alena Tesařová, atesarova@fit.vutbr.cz

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

10. týden

Organizační okénko

- Projekt č. 2
  - Odevzdání příští týden!
  - Všichni by už měli mít tým
- Zábavný experiment

Náplň cvičení

- ovládat dynamickou alokaci na hromadě (malloc, free, realloc)
- ovládat ladění pomocí debuggeru (gdb, valgrind)

Ladění pomocí gdb

1. Přeložte program pomocí:

```
gcc -std=c99 -g buggy.c -o buggy
```

2. Spusťte gdb:

```
(gdb) gdb buggy
```

3. Nastavte breakpoint na funkci main:

```
(gdb) break main
```

4. Spusťte program s argumenty:

```
(gdb) run arg1 arg2 arg3
```

Ladění pomocí gdb

Pomocí příkazů next, step, finish, cont procházejte průběhem programu:

(gdb) next - (nebo jen n) proveď další řádek programu
(gdb) step - (nebo jen s) proveď další krok (do) podprogramu
(gdb) finish - (nebo je fin) spusť funkci do jejího konce
(gdb) cont - pokračuj ve spuštění programu
(gdb) list - (nebo jen l) zobraz 10 dalších řádků kódu blízko aktuálnímu.

Tisk/Zobrazení hodnoty:

```
print/display EXPR
```

Ukončení gdb:

```
(gdb) quit
```

Valgrind

- Spuštění

```
valgrind ./2_replace
```

- Pro detailní popis chyb:

```
valgrind --leak-check=full ./2_replace
```

Kolik bylo ztraceno bytů?

```
atesarova@merlin: ~/malloc$ valgrind ./malloc
==16095== Memcheck, a memory error detector
==16095== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==16095== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==16095== Command: ./malloc
==16095==
Zadej cislo: 12
Nacteno cislo: 12
==16095==
==16095== HEAP SUMMARY:
==16095==     in use at exit: 4 bytes in 1 blocks
==16095==   total heap usage: 1 allocs, 0 frees, 4 bytes allocated
==16095==
==16095== LEAK SUMMARY:
==16095==    definitely lost: 4 bytes in 1 blocks
==16095==    indirectly lost: 0 bytes in 0 blocks
==16095==      possibly lost: 0 bytes in 0 blocks
==16095==    still reachable: 0 bytes in 0 blocks
==16095==         suppressed: 0 bytes in 0 blocks
==16095== Rerun with --leak-check=full to see details of leaked memory
==16095==
==16095== For counts of detected and suppressed errors, rerun with: -v
==16095== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Malloc a realloc

Jak alokujeme paměť pro řetězec “Hello” ?

```
char *ret = (char *)malloc((strlen(“Hello”) + 1)*sizeof(char))
```

Malloc a realloc

Co když v půlce programu si uvědomíme, že budeme potřebovat více paměti např. pro “Hello world”?

```
char *new_ret = realloc(ret, strlen(”Hello world”) + 1);
If (new_ret == NULL){
   // nepovedlo se alokovat více paměti, ale POZOR starý
   pointer ret je pořád platný a musí být uvolněn pomocí
   free()
   free(ret);
   return;
}
// povedlo se realokovat, POZOR ret je invalidní v tento
   moment, new_ret pak musí být uvolněn
printf(”%c”, ret[2]); // nelze
free(ret) // co by se stalo?
free(new_ret);
```

Příklad č. 1

Implementujte funkci pro realokaci dynamicky alokovaného pole

```
int vector_add(Vector *vec, int value);
```

```
// Vector representation
typedef struct
{
   int *items;
   unsigned int size;
} Vector;
```

```
Vector *vector_ctor() // konstruktor, size = 0 (items = NULL)
void vector_dtor(Vector *vector) // destructor, free zde
void vector_print_reverse(Vector vector)
```

Příklad č. 2 – kostra na wiki

1. Implementujte funkci pro nalezení podřetězce v řetězci.

```
int find_substr(char *str, char *substr)
```

2. Implementujte funkci pro nahrazení podřetězce stejně velkým, ale jiným řetězcem.

```
void replace_same_length(char *str, char *substr, char
   *new_substr)
```

3. Zobecněte funkci pro nahrazení podřetězce (jinak velkým) řetězcem

4. POZOR, na wiki jsou chyby, dokážete je najít a opravit všechny? (hint: VALGRIND)

Příklad

Příklad:

```
char str[] = "kapesnik";
char substr[] = "pes";

int find_substr(char *str, char *substr) => 2

char substr[] = "lak";
void replace_same_length(char *str, char *substr, char
   *new_substr) => *new_substr = "kalaknik"

char substr[] = "lako";
void replace(char *str, char *substr, char *new_substr) =>
   *new_substr = "kalakonik"

char substr[] = "la";
void replace(char *str, char *substr, char *new_substr) =>
   *new_substr = "kalanik"
```

Příklad č. 3 – bonusový viz wiki

Děkuji za pozornost