Linköpings Universitet
Institutionen för datavetenskap (IDA)
UPP-gruppen 2019-08-29

Kort guide till Valgrind

Valgrind är ett stort verktyg som vi kan använda för att upptäcka minnesläckor i de program vi skapat. Nedan följer en kort guide för hur vi anväder Valgrind. (Om du vill läsa på mer om Valgrind rekommenderar jag att besöka deras hemsida, www.valgrind.org.)

Steg 1: Kompilering

Vi har ett program som vi vill undersöka för minnesläckor. När vi kompilerar programmet behöver vi lägga till flaggan -g. Detta gör att vi får mer informion om var minnesläckor finns. I övrigt sker kompilering på exakt samma sätt:

```
g++ -std=c++17 -g test.cc
```

(OBS, test.cc är endast ett exempel. Ni ska kompilera den/de filer som ni vill köra och hitta minnesläckor i.)

Steg 2: Att köra Valgrind

Vi har kompilerat den/de filer vi vill köra och av det har det skapats en körbar fil (a.out om inget annat specificerats).

För att undersöka om programmet innehåller minnesläckor skriver vi följande i terminalen:

```
valgrind --tool=memcheck --leak-check=yes ./a.out
```

Där a.out är namnet på den körbara filen och ./ hänvisar till att den körbara filen finns i den mappen vi står i.

Valgrind kommer nu skriva ut information om ditt program. Nu behöver vi bara lista ut hur vi ska tolka det.

Steg 3.a: Minnesläckor finns

Följande program är ett exempel där vi har en minnesläcka. På rad 8 i programmet, under main, har vi allokerat minne för ett objekt av klassen A. Detta minnet lämnas sedan inte tillbaka till programmet.

test.cc:

```
class A{
  int x;
};

int main()
{
   A * a = new A();
   return 0;
}
```

Efter att vi kompilerat programmet och kört det med Valgrind kommer följande skrivas ut:

```
==18420== Memcheck, a memory error detector
==18420== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==18420== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==18420== Command: ./a.out
==18420==
==18420==
==18420== HEAP SUMMARY:
==18420==     in use at exit: 4 bytes in 1 blocks
==18420==   total heap usage: 2 allocs, 1 frees, 72,708 bytes allocated
==18420==
==18420== 4 bytes in 1 blocks are definitely lost in loss record 1 of 1
==18420==    at 0x4C3017F: operator new(unsigned long) (in /usr/lib/valgrind/vgpre
==18420==    by 0x10868B: main (test.cc:8)
==18420==
==18420== LEAK SUMMARY:
==18420==    definitely lost: 4 bytes in 1 blocks
==18420==    indirectly lost: 0 bytes in 0 blocks
==18420==      possibly lost: 0 bytes in 0 blocks
==18420==    still reachable: 0 bytes in 0 blocks
==18420==         suppressed: 0 bytes in 0 blocks
==18420==
==18420== For counts of detected and suppressed errors, rerun with: -v
==18420== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

I all denna text säger Valgrind oss två viktiga saker:

1. Att på rad 8, i filen test.cc, i funktionen main, finns det en minnesläcka.

```
==18420==    by 0x10868B: main (test.cc:8)
```

2. Att vi har tappat bort 4 bytes, som det står summerat i LEAK SUMMARY:

```
==18420== LEAK SUMMARY:
==18420==    definitely lost: 4 bytes in 1 blocks
==18420==    indirectly lost: 0 bytes in 0 blocks
==18420==      possibly lost: 0 bytes in 0 blocks
==18420==    still reachable: 0 bytes in 0 blocks
==18420==         suppressed: 0 bytes in 0 blocks
```

Steg 3.b: Minnesläckor finns inte

Följande program är ett exempel där vi har en minnesläcka. Nu ger vi tillbaka det minnet som vi allokerat, genom att utföra delete på pekaren.

test.cc:

```
class A{
  int x;
};

int main()
{
  A * a = new A();
  delete a;
  return 0;
}
```

Efter att vi kompilerat programmet och kört det med Valgrind kommer följande skrivas ut:

```
==13369== Memcheck, a memory error detector
==13369== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==13369== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==13369== Command: ./a.out
==13369==
==13369==
==13369== HEAP SUMMARY:
==13369==     in use at exit: 0 bytes in 0 blocks
==13369==   total heap usage: 2 allocs, 2 frees, 72,708 bytes allocated
==13369==
==13369== All heap blocks were freed -- no leaks are possible
==13369==
==13369== For counts of detected and suppressed errors, rerun with: -v
==13369== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Om det inte finns några läckor kommer det att stå:

```
==13369== All heap blocks were freed -- no leaks are possible
```