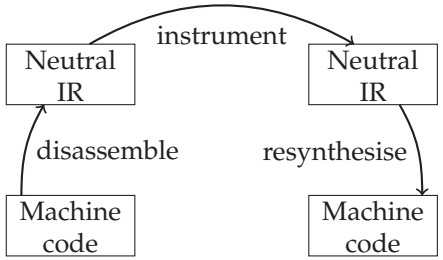
Port of Valgrind to Solaris/x86

Author: Ing. Petr Pavlů (FIT ČVUT), Advisor: Mgr. Jiří Svoboda (Oracle Czech)

Valgrind

Valgrind is a dynamic instrumentation framework and a set of associated tools which can detect many programming errors and also do profiling. It is a free software available for Linux, FreeBSD and Mac OS X.
It uses disassemble-and-resynthesize instrumentation with an architecture-neutral intermediate representation.



Solaris/x86

Solaris is a commercial Unix-type operating system developed by the Oracle Corporation. The port is focused on its Intel x86-32 variant.

Memcheck Example

A problem found by the Memcheck tool in the Solaris standard C library:

```
setup@sol:~$ cat bug.c
#include <stdio.h>
int main(void)
{
  char buf[64];
  snprintf(buf, sizeof(buf), "Hello");
  return 0;
}
setup@sol:~$ cc -g bug.c -o bug
setup@sol:~$ valgrind --quiet --track-origins=yes ./bug
==857== Conditional jump or move depends on uninitialised
        value
==857==    at 0xFEF20AAF: getxfdat (in /lib/libc.so.1)
==857==    by 0xFEF20B47: _realbufend (in /lib/libc.so.1)
==857==    by 0xFEF0FB7A: _ndoprnt (in /lib/libc.so.1)
==857==    by 0xFEF1446D: snprintf (in /lib/libc.so.1)
==857==    by 0x8050CC0: main (bug.c:5)
==857==  Uninitialised value was created by a stack
         allocation
==857==    at 0xFEF1440C: snprintf (in /lib/libc.so.1)
```

A hack is used inside the Solaris standard C library to differentiate between two structures. The ported Memcheck tool correctly detects it as a potential problem.

Port Implementation

Valgrind is tightly tied to an underlying operating system, therefore porting it to a new platform is a challenging task that requires detailed knowledge of the targeted environment.
The changes needed to support Solaris/x86 in Valgrind span over several areas:

- build system,
- Valgrind's standard library,
- client program loading,
- thread support,
- signal processing,
- system calls (Valgrind needs to know effects of each syscall),
- debug information reader,
- function replacement.

The functionality of the port and the tools was evaluated using the Valgrind test suite (consisting of more than 300 tests), proving that the port works correctly.

Program Analysis

Program analysis is a process of automatically analysing behaviour of computer programs. It can be split into two categories: static and dynamic analysis. Static analysis derives properties of programs without executing them. Dynamic analysis observes programs by executing them. Valgrind is a representative of the dynamic approach.

Cachegrind Example

Profiling branch prediction of the GNU C compiler:

```
setup@sol:~$ valgrind --tool=cachegrind --cache-sim=no
  --branch-sim=yes cc bug.c
==854== I   refs:      1,118,378
==854==
==854== Branches:        248,769  (245,051 cond + 3,718 ind)
==854== Mispredicts:      18,625  ( 18,273 cond +   352 ind)
==854== Mispred rate:        7.4% (    7.4%     +   9.4%   )
```

Tools

Valgrind is shipped with eleven official tools. The port supports nine of them, including Memcheck (a memory error detector), Cachegrind (a cache and branch prediction profiler) and Massif (a heap profiler).
Only two thread error detectors (DRD and Helgrind) are currently not available.