IBM's Migration Kit for Solaris OS to Linux

Dr. Wolfgang Gellerich
gellerich@de.ibm.com

February 2007, Session 9256

Agenda

- Overview – purpose of the Migration Kit and its contents
- Technical Documentation provided with the Migration Kit
- Development tools provided with the Migration Kit
- Sizing a migration project
- How to obtain the Migration Kit
- Trademark notices

The Migration Kit for Solaris OS to Linux

- Migrating applications from Solaris OS to Linux requires detailed knowledge about differences concerning
  - libraries and operating system interfaces,
  - the development environment, including compiler, tools for building whole projects, managing source code repositories, packaging software,
  - system administration.
- The Migration Kit provides information about all these topics.
  - Three tools analyze source code and provide assistance for making adaptations.
  - Two text documents address administrative issues.

What is provided with the Migration Kit ?

- Interactive tools to assist in porting applications:
  - Source Checking Tool
    Detects Solaris-specific constructs in C and C++ sources
    Assesses porting effort
  - Endian Checking Tool
    Identifies endian issues in C and C++ sources
  - Shell Script Checking Tool
    Identifies OS-specific differences in shell scripts
- Technical documentation (PDF files):
  - The “Guide to Application Porting from Solaris OS to Linux”
  - The IBM Redbook “Solaris to Linux Migration: A Guide for System Administrators”
  - Documentation for all tools

The “Guide to Application Porting From Solaris OS to Linux”

- Recommendation how to best organize a migration project
- Technical differences concerning the development environment, including:
  - make
  - compiler
  - linker
- Architecture-specific differences, including
  - Sizes of base data types and their alignment, 32 to 64 bit migration
  - Endian-ness (supported by the Endian Checking Portability Tool)
  - System call and library functions (supported by the Code Checking Tool)
- Performance tuning tools available for Linux
- Software packaging tools available for Linux

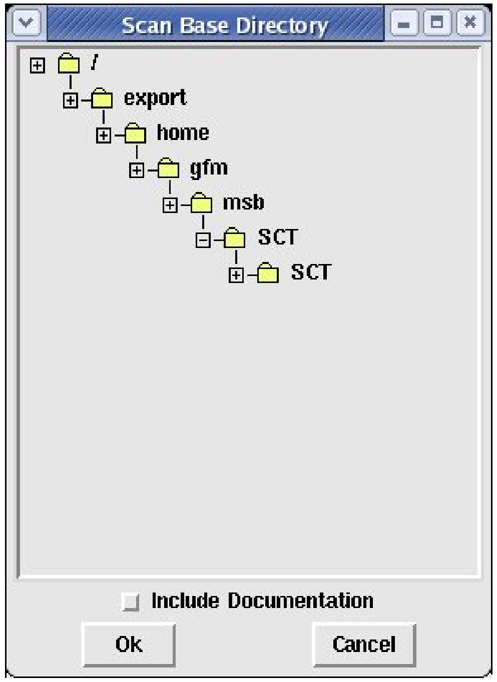
The “Guide for System Administrators”

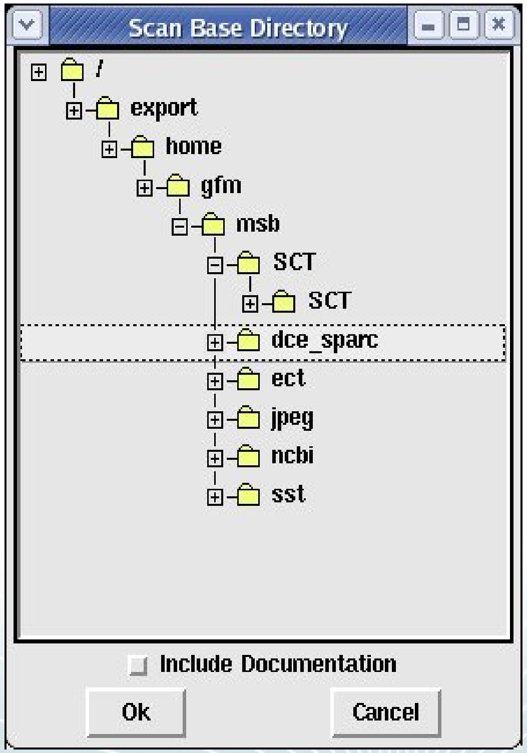
- Provides task-based grouping of differences between the two operating systems
- Covers topics like:
  - Operating system installation, initialization and booting
  - Disk, file system and device management
  - Printing
  - Networks
  - Users and groups
  - Security issues

The Source Checking Tool 1/6

- Input: Solaris OS application source code, written in C or C++
- Reports code patterns specific to Solaris OS:
  - Calls to the Solaris OS Application Program Interface (API)
  - Include files specific to Solaris OS
  - Sun compiler pragmas
  - Suggests Linux alternatives
- Knowledge database:
  - More than 3800 database entries in total
  - 624 text files providing technical documentation
- Output:
  - GUI offers interactive access
  - Annotated source code files available

The Source Checking Tool 2/6



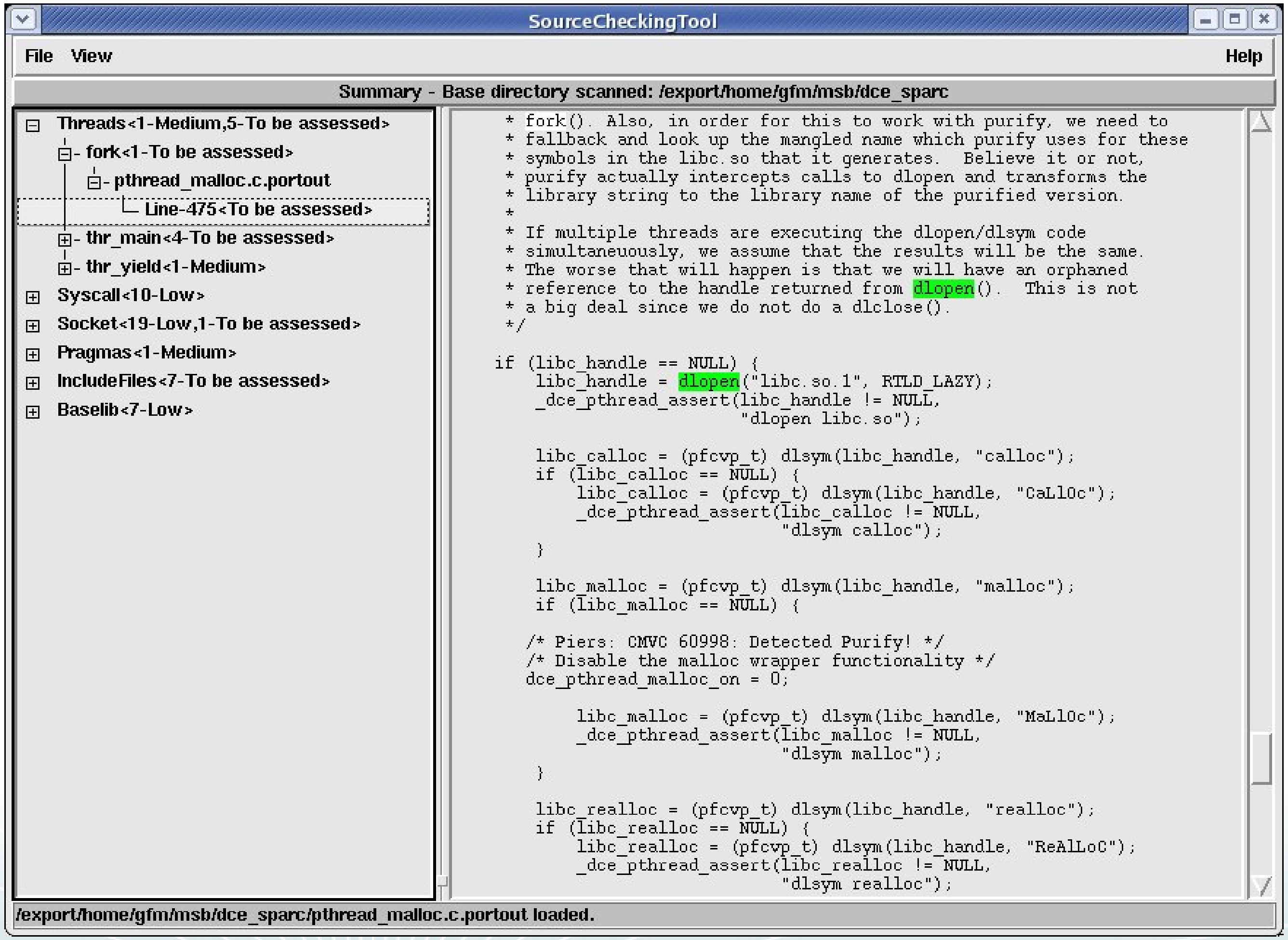


This tool goes over C/C++ files, or directories of C/C++ files, to find potential problems with API or compiler pragma usage.

You can use a GUI to select which file or directory to process.

The Source Checking Tool 3/6

When the scan is complete, you may inspect each item discovered, file by file.



Summary - Base directory scanned: /export/home/gfm/msb/dce_sparc

```
⊟ Threads<1-Medium,5-To be assessed>
   ⊟- fork<1-To be assessed>
      ⊟- pthread_malloc.c.portout
         └ Line-475<To be assessed>
   ⊞- thr_main<4-To be assessed>
   ⊞- thr_yield<1-Medium>
⊞ Syscall<10-Low>
⊞ Socket<19-Low,1-To be assessed>
⊞ Pragmas<1-Medium>
⊞ IncludeFiles<7-To be assessed>
⊞ Baselib<7-Low>
```

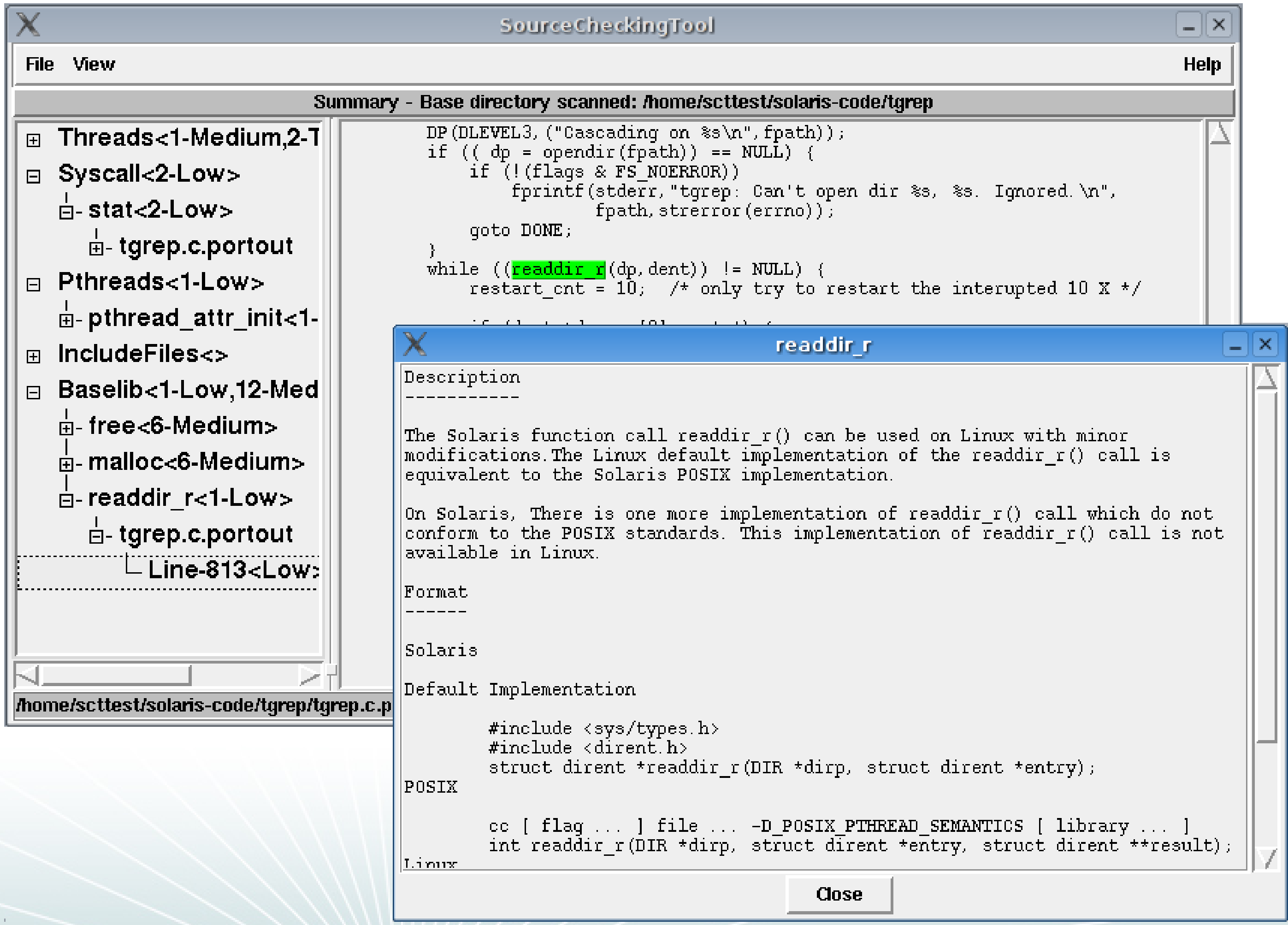
```
     * fork(). Also, in order for this to work with purify, we need to
     * fallback and look up the mangled name which purify uses for these
     * symbols in the libc.so that it generates.  Believe it or not,
     * purify actually intercepts calls to dlopen and transforms the
     * library string to the library name of the purified version.
     *
     * If multiple threads are executing the dlopen/dlsym code
     * simultaneuously, we assume that the results will be the same.
     * The worse that will happen is that we will have an orphaned
     * reference to the handle returned from dlopen().  This is not
     * a big deal since we do not do a dlclose().
     */

    if (libc_handle == NULL) {
        libc_handle = dlopen("libc.so.1", RTLD_LAZY);
        _dce_pthread_assert(libc_handle != NULL,
                            "dlopen libc.so");

        libc_calloc = (pfcvp_t) dlsym(libc_handle, "calloc");
        if (libc_calloc == NULL) {
            libc_calloc = (pfcvp_t) dlsym(libc_handle, "CaLl0c");
            _dce_pthread_assert(libc_calloc != NULL,
                                "dlsym calloc");
        }

        libc_malloc = (pfcvp_t) dlsym(libc_handle, "malloc");
        if (libc_malloc == NULL) {

       /* Piers: CMVC 60998: Detected Purify! */
       /* Disable the malloc wrapper functionality */
       dce_pthread_malloc_on = 0;

            libc_malloc = (pfcvp_t) dlsym(libc_handle, "MaLl0c");
            _dce_pthread_assert(libc_malloc != NULL,
                                "dlsym malloc");
        }

        libc_realloc = (pfcvp_t) dlsym(libc_handle, "realloc");
        if (libc_realloc == NULL) {
            libc_realloc = (pfcvp_t) dlsym(libc_handle, "ReAlLoC");
            _dce_pthread_assert(libc_realloc != NULL,
                                "dlsym realloc");
```

/export/home/gfm/msb/dce_sparc/pthread_malloc.c.portout loaded.

The Source Checking Tool 4/6

The tool provides information about necessary changes.

The tool knows details of over 3800 different system calls, library functions, and compiler pragmas.

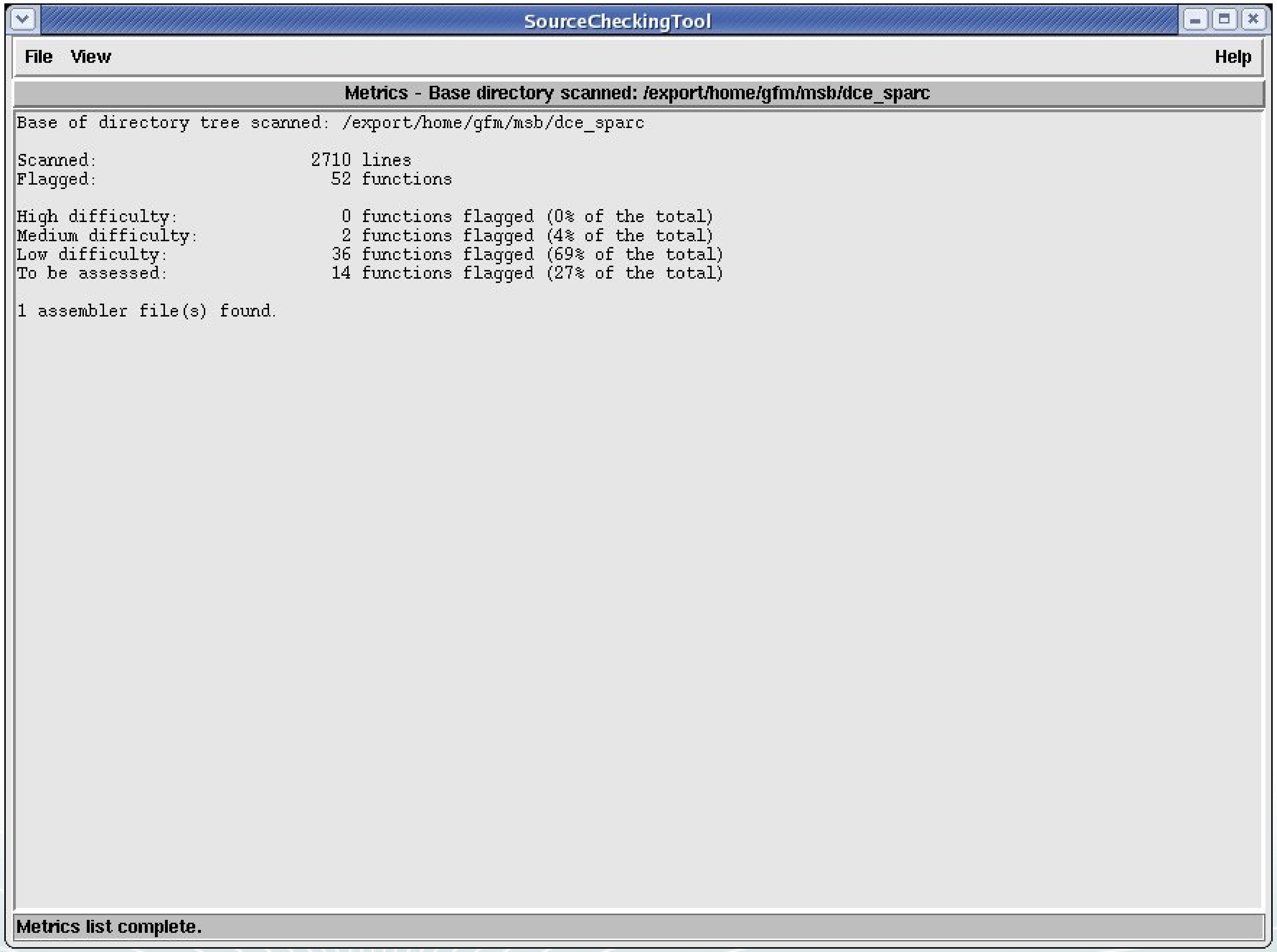


```
        DP(DLEVEL3,("Cascading on %s\n",fpath));
        if (( dp = opendir(fpath)) == NULL) {
            if (!(flags & FS_NOERROR))
                fprintf(stderr,"tgrep: Can't open dir %s, %s. Ignored.\n",
                        fpath,strerror(errno));
            goto DONE;
        }
        while ((readdir_r(dp,dent)) != NULL) {
            restart_cnt = 10;  /* only try to restart the interupted 10 X */
```

```
Description
-----------

The Solaris function call readdir_r() can be used on Linux with minor
modifications.The Linux default implementation of the readdir_r() call is
equivalent to the Solaris POSIX implementation.

On Solaris, There is one more implementation of readdir_r() call which do not
conform to the POSIX standards. This implementation of readdir_r() call is not
available in Linux.

Format
------

Solaris

Default Implementation

        #include <sys/types.h>
        #include <dirent.h>
        struct dirent *readdir_r(DIR *dirp, struct dirent *entry);
POSIX

        cc [ flag ... ] file ... -D_POSIX_PTHREAD_SEMANTICS [ library ... ]
        int readdir_r(DIR *dirp, struct dirent **entry, struct dirent **result);
```

The Source Checking Tool 5/6

The results are available in terms of project metrics.



Metrics - Base directory scanned: /export/home/gfm/msb/dce_sparc

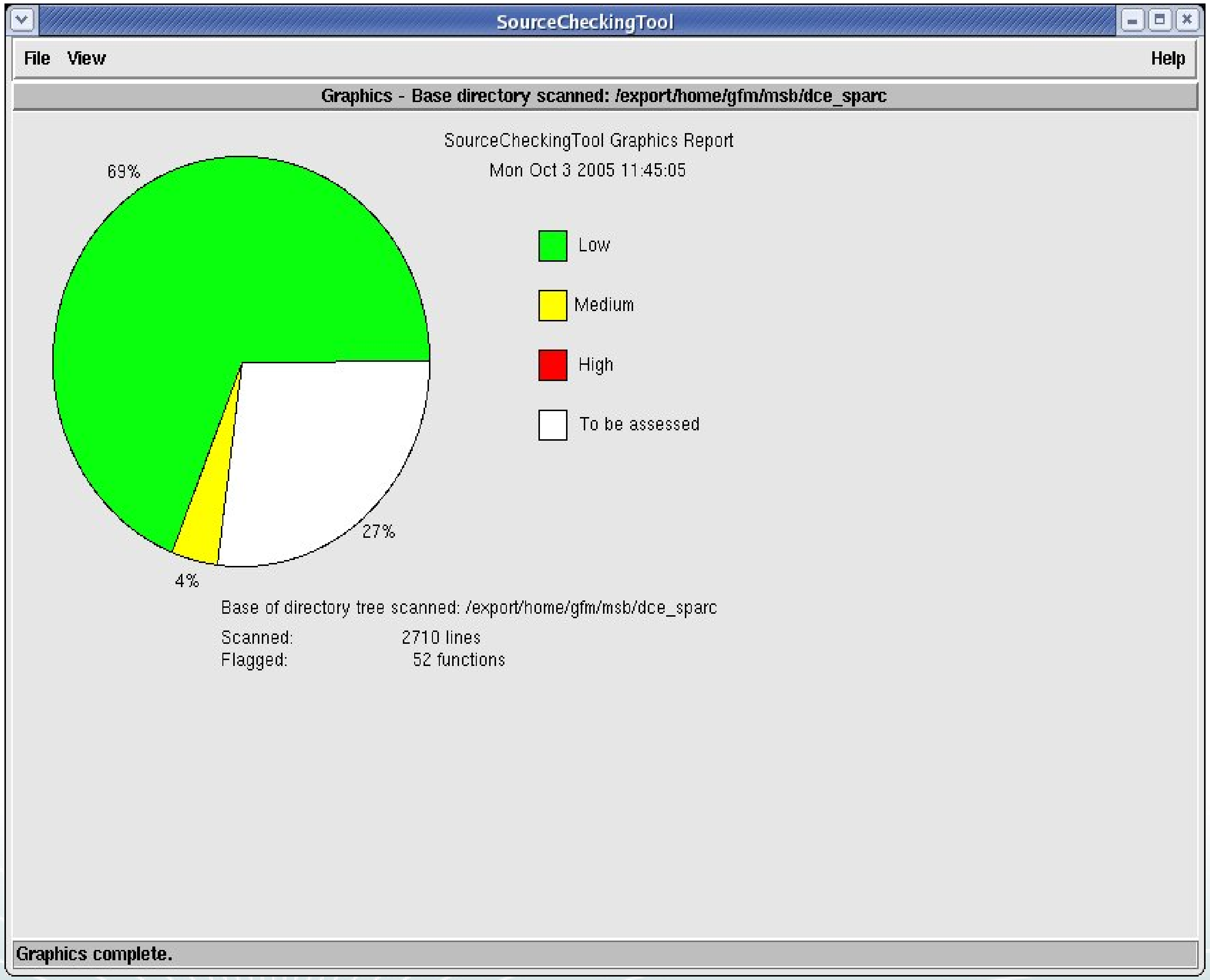
```
Base of directory tree scanned: /export/home/gfm/msb/dce_sparc

Scanned:                     2710 lines
Flagged:                       52 functions

High difficulty:                0 functions flagged (0% of the total)
Medium difficulty:              2 functions flagged (4% of the total)
Low difficulty:                36 functions flagged (69% of the total)
To be assessed:                14 functions flagged (27% of the total)

1 assembler file(s) found.
```

Metrics list complete.

The Source Checking Tool 6/6

A graphical view is also available.

Graphics can be saved for further usage (include into presentations etc.).



SourceCheckingTool Graphics Report
Mon Oct 3 2005 11:45:05

Base of directory tree scanned: /export/home/gfm/msb/dce_sparc

```
Scanned:                2710 lines
Flagged:                  52 functions
```

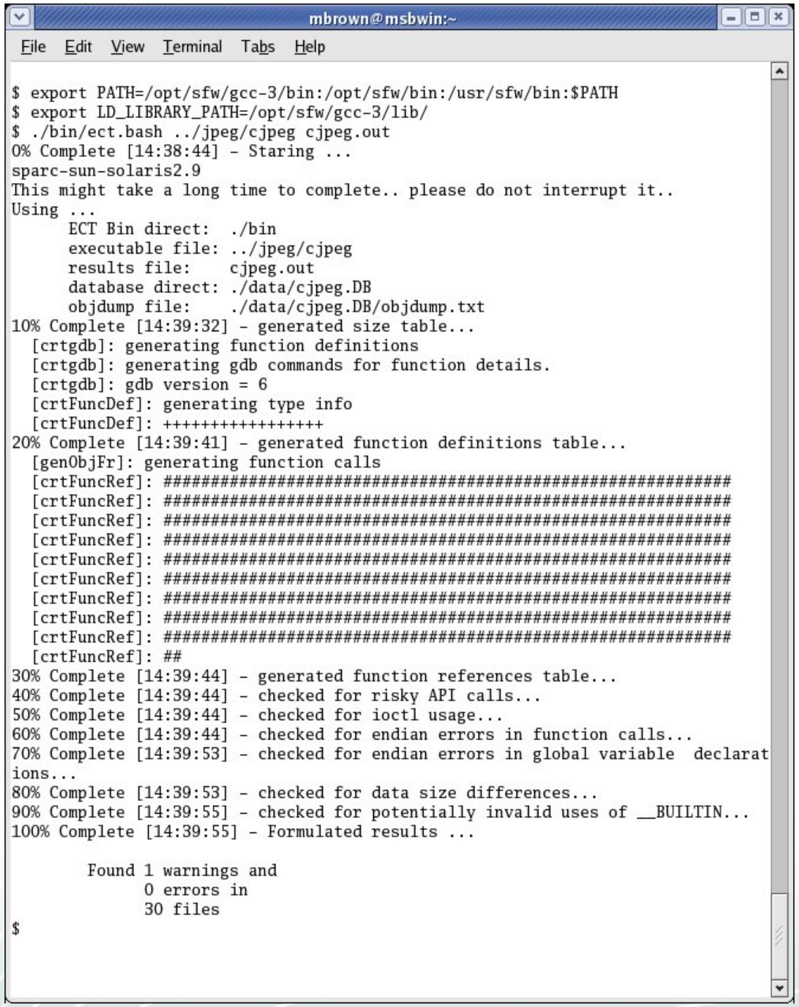
The Endian Checking Tool 1/3

- Detects code patterns that may cause an endian problem
- Example: several orders to store binary value 4A3B2C1D
  - Big-Endian 0x4A 0x3B 0x2C 0x1D
  - Little-Endian 0x1D 0x2C 0x3B 0x4A
- Affected code patterns include:
  - Byte-oriented processing of binary data
  - Data structures accessed by assembler code
  - Type conversion by (mis-)using pointers or union
- Input is a combination of:
  - Source Code
  - Binary compiled with profiling options enabled

The Endian Checking Tool 2/3

Before the tool can analyze the code, it must first be compiled using the usual Linux compiler.

Once the code is prepared, the tool can be used on a command-line or GUI basis.



```
$ export PATH=/opt/sfw/gcc-3/bin:/opt/sfw/bin:/usr/sfw/bin:$PATH
$ export LD_LIBRARY_PATH=/opt/sfw/gcc-3/lib/
$ ./bin/ect.bash ../jpeg/cjpeg cjpeg.out
0% Complete [14:38:44] - Staring ...
sparc-sun-solaris2.9
This might take a long time to complete.. please do not interrupt it..
Using ...
      ECT Bin direct:  ./bin
      executable file: ../jpeg/cjpeg
      results file:    cjpeg.out
      database direct: ./data/cjpeg.DB
      objdump file:    ./data/cjpeg.DB/objdump.txt
10% Complete [14:39:32] - generated size table...
  [crtgdb]: generating function definitions
  [crtgdb]: generating gdb commands for function details.
  [crtgdb]: gdb version = 6
  [crtFuncDef]: generating type info
  [crtFuncDef]: ++++++++++++++++
20% Complete [14:39:41] - generated function definitions table...
  [genObjFr]: generating function calls
  [crtFuncRef]: ############################################################
  [crtFuncRef]: ############################################################
  [crtFuncRef]: ############################################################
  [crtFuncRef]: ############################################################
  [crtFuncRef]: ############################################################
  [crtFuncRef]: ############################################################
  [crtFuncRef]: ############################################################
  [crtFuncRef]: ############################################################
  [crtFuncRef]: ############################################################
  [crtFuncRef]: ##
30% Complete [14:39:44] - generated function references table...
40% Complete [14:39:44] - checked for risky API calls...
50% Complete [14:39:44] - checked for ioctl usage...
60% Complete [14:39:44] - checked for endian errors in function calls...
70% Complete [14:39:53] - checked for endian errors in global variable  declarat
ions...
80% Complete [14:39:53] - checked for data size differences...
90% Complete [14:39:55] - checked for potentially invalid uses of __BUILTIN...
100% Complete [14:39:55] - Formulated results ...

        Found 1 warnings and
              0 errors in
              30 files
$
```

The Endian Checking Tool 3/3

- Typical finding: parameter size mismatch

```
/test/src/init.c – Line 199: E30001
Variable/parameter size mismatch arg 2 size 4
in call to mystrncpy. (Defined in
/test/src/init.c at line 190 size 1)
```

- Formal parameter declaration: type has size of 4 bytes
- The actual argument has a size of 1 byte
- Where is this byte stored within the four bytes available?

The Shell Script Checking Tool

- Examines shell scripts looking for:
  - Path issues,
  - File issues,
  - Utility programs
- Covers:
  - Bourne,
  - csh,
  - ksh,
  - and variants
- ...and provides recommendations on what changes might be necessary.

Shell Script Checking Tool Example

```
+++ Begin report for 'autofs' Mon Aug  1 09:12:00 EDT 2005

+++ Summary Information for 'autofs'
4  Total Items identified

1  E1001 Items: File path does not exist on Linux
2  E4001 Items: Directory structure or file path may not map directly on Linux
1  E7001 Items: Comparable Linux Command may exist

+++ Possible Error Code Resolutions
E1001 Consult Linux man pages to determine if a comparable Linux path exists for these
   files.
E4001 Consult Linux man pages to determine the path the script should use.

+++ Detail Information for autofs
Line       Item                Error code and Message
-------------------------------------------
  12       '/dev'           E4001 Files under this Path may not map directly on Linux
  12       '/lib'           E4001 Files under this Path may not map directly on Linux
  12       '/usr/lib/autofs/automountd'  E1001 File path does not exist on Linux
  17       'umountall'      E7001 comparable Linux command might be  "umount -a"

+++ End of report for 'autofs'
```

Sizing a Migration Project 1/2

- A rough classification of the entries in the Source Checking Tool's knowledge database:
  - 46 % of all calls are identical in Linux and Solaris
    e.g. mathematical functions found in math.h
  - 8 % require trivial changes
    e.g. name of a function is different
  - 6 % require changes in local program context
    e.g. different order of function arguments
  - 15 % require major non-local changes
    ...in case of different semantics
  - 25 % need to be assessed in application context
- Result from one large real-world project:
  Source Checking Tool reported one finding / 400 LOC

Sizing a Migration Project 2/2

- What is the effort for a migration project ?
  - Probably considerably less than its initial development
  - Actual cost will strongly depend on the individual project
- Migration effort strongly depends on the portability of the source code:
  - Relying on standardized libraries only will reduce the effort
  - Proper program organization and modular structuring will reduce the effort
  - Using libraries specific for a proprietary OS will cost additional effort
  - Performance optimizations based on particular properties of an OS will cost additional effort

Supported Versions

- Operating system versions covered by the Migration Kit:
  - Solaris versions: 8 and 9
  - Linux Kernel 2.6
  - Including libraries and compilers usually used with these versions
- The tools provided with the Migration Kit will work on computers running Solaris version 8 or 9
- The format of the included documentation is PDF

How to obtain the Migration Kit for Solaris OS to Linux

The toolkit is available free of charge from the following URL:

`http://www-1.ibm.com/partnerworld/pwhome.nsf/weblook/pat_linux_migrate_solaris.html`

Trademarks 1/2

Trademarks 2/2

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.
Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.
Microsoft, Windows, Windows NT and the Windows logo are registered trademarks of Microsoft Corporation in the United States and/or other countries.
Intel, Itanium and Pentium are registered trademarks and Intel Xeon and MMX are trademarks of Intel Corporation in the United States and/or other countries
AMD Opteron is a trademark of Advanced Micro Devices, Inc.
TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).
SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECapc, SPEChpc, SPECjvm, SPECmail, SPECimap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).
NetBench is a registered trademark of Ziff Davis in the United States, other countries or both.
Sun, Sun Microsystems, the Sun logo, iForce, Java, Netra, N1, Solaris, Sun Fire, Sun Ray, SunSpectrum, Sun StorEdge, SunTone, The Network is the Computer, all trademarks and logos that contain Sun, Solaris, or Java, and certain other trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.
All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

Other company, product and service names may be trademarks or service marks of others.