



CountCat User Guide

Installation and Migration Guide

2009.08.19
Version 1.3a

FUJITSU Migration Services

Contents

Introduction.....	4
Reasons for Migration.....	4
CountCat Installation.....	5
Brief Description.....	5
Supported Operating Systems.....	5
Installation.....	5
Invoking CountCat.....	5
ccat Command Line Options.....	6
Script Flow.....	7
Features/Bugs.....	7
Selecting Source and Target Systems.....	8
Source File Audit.....	8
File Audit.....	8
Header File Analysis.....	9
Code Analysis.....	9
System Calls.....	9
Solaris threads and POSIX pthreads.....	12
LinuxThreads.....	13
Other Thread Differences.....	14
Common Solaris Threads with Linux Alternatives.....	14
Common Solaris Threads Without Linux Alternatives.....	15
Mutex Threads.....	15
Semaphores.....	15
Semaphore Concepts.....	15
Linux Compatible.....	16
Solaris Only Semaphores.....	16
Sockets.....	16
Running CountCat.....	17
Types of Code Audit.....	19
Full Audit Example.....	20
CountCat output Files.....	21
ccat.html.....	21
ccat_detail.html.....	23

Update History

Edition	Date	What change/Reason	Changes	Author
Pre 1.0	2008.10.31	Halloween Release - Pretty Scary	Just started	Andy Prowse
Pre 1.1	2008.11.14	Halfway there	Added information	Andy Prowse
Pre 1.2	2008.12.04	Added screen shots	Added information	Andy Prowse
1.0	2008.12.11	Minor changes, first release	Added information	Andy Prowse
1.1	2009.01.09	Format fix		Andy Prowse
1.2	2009.05.26	Updated to match version FMSccat 1.0.5	Added functionality	Andy Prowse
1.3	2009.06.01	Milestone release corresponding with FMSccat 1.3.0 universal script	Added functionality and universal	Andy Prowse
1.3a	2009.08.19	Fixing various grammar/spelling issues		Andy Prowse

References:

No.	Document	Author
1	Solaris Man Pages	Sun Microsystems
2	Linux Man Pages	RedHat, Many Others
3	The GNU C Library Reference Manual	S. Loosemoore

Introduction

This document is intended to aid in the migration of SPARC Solaris specific source code to an Intel x86 based environment utilizing Linux.

We hope that the CountCat User Guide will assist in the analysis of source code, and make the code porting easier. With CountCat, and your compiler, finding areas that require modification for linux should be easy to spot. Always use the latest version of CountCat as it is continually being refined and improved.

Happy Porting,
The CountCat Development Team.

Background

Reasons for Migration

There are many good reasons to migrate an application or system to a new platform. Some will make sense for your organization, others will not. There are some applications that are best suited to stay on a SPARC system, either for economic, hardware/software or performance reasons. That is perfectly fine. What should be addressed are the systems that make economic sense to move, be that economic reason is financial, environmental or footprint.

With x86 hardware development outpacing current SPARC technology, many benefits can be passed to the data centre. Even older applications that are locked into a historical release of Solaris can be migrated to newer x86 technologies with significant gains in performance and reliability through emulation software.

CountCat Installation

Brief Description

CountCat performs a quick and dirty analysis of lines of SPARC Solaris C-source code and tries to determine the effort required in porting to a Linux based system. During the analysis of the source code, countcat will flag any areas that may be of concern during the porting of the code.

Supported Operating Systems

For Solaris, both SPARC and x86 is supported.

For Linux, SUSE, RedHat and Ubuntu are supported. Any other Linux that uses the ksh shell should also have no issues.

Installation

To install countcat, as root expand and untar the FMSccat tarball in the / directory. This will install the countcat script into the /opt/FMSccat directory. Under the /opt/FMSccat/bin directory, you will find the ccat script.

Invoking CountCat

Once the installation is complete, run ccat in the root of the source tree.

On the first run of the ccat script, it will ask for the source system OS level, and also the OS of the target system.

Example:

```
# cd [source_code_dir]
# /opt/FMSccat/bin/ccat [quick|script|clean|reset|help]
```

During the execution of the script, ccat will provide a quick summary on the console. In addition, countcat will leave an html formatted report, and ccat.html, a more in depth, detailed report called ccat_detail.html. For easier data management, you will also find a CSV file called ccat.csv. The CSV file uses “---” as delimiters.

The CountCat generated reports, in addition to this migration guide, will help to identify the occurrences and levels of difficulty of any possible issues on migrating code from SPARC Solaris to x86 Linux.

ccat is to be used in conjunction with the Migration Guide.

ccat Command Line Options

ccat has five command line options:

help

Help will display a brief description of the command line options.

quick

Quick will give an inventory of C code files and header files, along with a count of total lines.

script

Script will inventory all the scripts that are located in the source code directory, and itemize them in order of type.

clean

ccat will restore itself back to it's initial `install fresh` condition before running. This allows for selecting which release of Solaris the source code is coming from, and to which release of Linux the code is being ported to.

reset

Reset is the same as clean, but does not launch the ccat script after cleaning.

No command line option will run ccat in full. This can take some time to execute.

Script Flow

The initial phase of ccat inventories the source code tree and itemizes the files and the total number of lines of both C-source and header files.

Next, the header files are compared and sorted as follows:

1. Header files that are common between both Solaris and Linux
2. Header files that are in Solaris only
3. Header files that are included in the source code

Following this, pattern matching is performed looking for Solaris only system calls, library calls, socket calls, POSIX and Solaris threads and semaphores. Also, binary calls, union declarations, terminal I/O error messages and other such code will be looked for.

Scripts are also searched, as they will have differences between the two platforms.

Features/Bugs

Currently ccat does not distinguish comment lines from code.

CountCat Flow

Selecting Source and Target Systems

When invoking countcat for the first time, it will prompt for the source and target system for the migration.

For the source system, you are provided with a choice of:

- Solaris 7
- Solaris 8
- Solaris 9
- Solaris 10
- Solaris 11

For the target system, the choices are as follows:

- RedHat Enterprise Linux 4
- RedHat Enterprise Linux 5
- SUSE Linux Enterprise Server 10
- Analyse for this system

At this point, the databases will be generated to match the source code to.

Source File Audit

File Audit

The initial phase of the audit will count the total number of C source code files and lines. A listing of the files, locations and lines per will be displayed in the ccat.html file. This is repeated again for the header files. A total of all lines and the number of files is then generated.

Header File Analysis

Once the files are audited, the analysis begins. The first step is to match the header files against the files supplied by the source Solaris system, the target Linux system and what is provided in the source code itself.

The first stage gives a total of unique header files that are referenced in the C source code.

The second stage lists the headers that included in the source code tree. These header files will need to be analysed for Linux compatibility.

The third stage lists the header files that common to both Solaris and Linux. At this point in time, they will need to be manually audited as the differences have yet to be documented.

The fourth stage lists the header files that are in Solaris only. Alternatives to these files will need to be located, or written to satisfy the requirements of the code. It may also be possible to modify the C code itself to work around any of these header issues.

Code Analysis

System Calls

These are system calls that are in Solaris, but not present in Linux.

Solaris System Call	Linux	Comment
acl	None	Get or set a file's Access Control List (ACL)
audit	None	Write a record to the audit log
auditon	None	Manipulate auditing
auditsvc	None	Write audit log to specified file descriptor
facl	None	Get or set a file's Access Control List (ACL)
fork1	None	Create a new process
fstatat	None	Get file status
getacct	None	Get, put, or write extended accounting data
getaudit	None	Get and set process audit information
getaudit_addr	None	Get and set process audit information
getauid	None	Get and set user audit identity
getprojid	None	Set or get task or project id
getrctl	None	Set or get resource control names
gettaskid	None	Set or get task or project id
_lwp_cond_broadcast	None	Signal a condition variable
_lwp_cond_realtimedwait	None	Wait on a condition variable
_lwp_cond_signal	None	Signal a condition variable

Solaris System Call	Linux	Comment
_lwp_cond_timedwait	None	Wait on a condition variable
_lwp_cond_wait	None	Wait on a condition variable
_lwp_info	None	Return the time accounting information of a single LWP
_lwp_kill	None	Send a signal to a LWP
_lwp_mutex_lock	None	Mutual exclusion
_lwp_mutex_trylock	None	Mutual exclusion
_lwp_mutex_unlock	None	Mutual exclusion
_lwp_self	None	Get LWP identifier
_lwp_sema_init	None	Semaphore operations
_lwp_sema_post	None	Semaphore operations
_lwp_sema_trywait	None	Semaphore operations
_lwp_sema_wait	None	Semaphore operations
memcntl	None	Memory management control
msgids	None	Discover all message queue identifiers
ntp_adjtime	None	Adjust local clock parameters
ntp_gettime	None	Get local clock values
pcsample	None	Program execution time profile
priocntl	None	Process scheduler control
priocntlset	None	Generalised process scheduler control
processor_bind	None	Bind LWPs to a processor
processor_info	None	Determine type and status of a processor
pset_assign	None	Manage sets of processors
pset_bind	None	Bind LWPs to a set of processors
pset_create	None	Manage sets of processors
pset_destroy	None	Manage sets of processors
pset_getattr	None	Get processor set attributes
pset_info	None	Get information about a processor set
pset_list	None	Get a list of processor sets
pset_setattr	None	Set processor set attributes
putacct	None	Get, put, or write extended accounting data
resolvepath	None	Resolve all symbolic links of a path name
semids	None	Discover all semaphore identifiers
setaskid	None	Set or get task or project id
setaudit	None	get and set process audit information
setaudit_addr	None	Get and set process audit information
setauid	None	Get and set user audit identity
setrctl	None	Set or get resource control names
shmids	None	Discover all shared memory identifiers
sigsend	None	Send a signal to a process or a group of processes
sigsendset	None	Send a signal to a process or a group of processes
__sparc_utrap_install	None	Install a SPARC V9 trap handler
swapctl	None	Manage swap space
uadmin	None	Administrative control
umount2	None	Unmount a file system
wracct	None	Get, put, or write extended accounting data
yield	None	Yield execution to another LWP

Different means that there are fundamental differences in the way that the system call functions between Solaris and Linux. Each of these calls need to be looked at , and any modifications made for the application to perform as anticipated.

Solaris System Call	Linux	Comment
futimesat	Different	Set file access and modification times
getdents	Different	Get directory entries
llseek	Different	Reposition read/write file offset
mount	Different	Mount and unmount a file system
openat	Different	Open a file
readv	Different	Read from a file
unlinkat	Different	Remove a directory entry
umount	Different	Mount and unmount a file system
utime	Different	Change access and/or modification times of an inode
utimes	Different	Change access and/or modification times of an inode
waitid	Different	Wait for a process to change state
writv	Different	Write to a file

Small Differences require the addition of another header file, or checking that the data passed to the call is in the correct format. The easiest way to determine the differences is to look an the man pages from both the source and target systems.

Solaris System Call	Linux	Comment
fstat	Small Differences	Get file status
getmsg	Small Differences	Get next message off a stream
getpmsg	Small Differences	Get next message off a stream
lstat	Small Differences	Get file status
msgctl	Small Differences	Message control operations
msgget	Small Differences	Get message queue
msgrcv	Small Differences	Message receive operation
msgsnd	Small Differences	Message send operation
open	Small Differences	Open or create a file
stat	Small Differences	Get file status

These System Calls require the `_BSD_SOURCE` macro to be defined - If this macro is defined, 4.3 BSD Unix functionality is included as well as ANSI C, POSIX.1 and POSIX.2. There are conflicts between some features of 4.3 BSD and POSIX.1. Enabling this macro caused the 4.3 BSD definitions to take priority over the POSIX.1 definitions.

Solaris System Call	Linux	Comment
adjtime	<code>_BSD_SOURCE</code>	Correct the time to allow synchronization of the system clock

These System Calls require the `_XOPEN_SOURCE=500` macro to be defined - Includes all definitions that are included by `_XOPEN_SOURCE_EXTENDED`.

Solaris System Call	Linux	Comment
pread	<code>_XOPEN_SOURCE=500</code>	Read from a file
pwrite	<code>_XOPEN_SOURCE=500</code>	Write on a file

Solaris threads and POSIX pthreads

POSIX and Solaris threads each have their own implementation of the threads library. The `libpthread` library is associated with POSIX; and the `libthread` library is associated with Solaris. Both implementations are interoperable, their functionality similar, and can be used with the same application. Only POSIX threads are guaranteed to be fully portable to other POSIX-compliant environments. POSIX and Solaris threads require different source, include files and linking libraries. It is this problem that will cause issues with the migration of the code from Solaris to Linux.

Most of the functions in the `libpthread` and `libthread` libraries have counterparts in the other corresponding library. POSIX function names, with the exception of the semaphore names, have `pthread` pre-pended to it. Function names for similar POSIX and Solaris have similar endings. Typically, similar POSIX and Solaris functions have the same number and use of arguments.

- POSIX pthreads and Solaris threads differ in the following ways:
- POSIX pthreads are more portable.
- POSIX pthreads establish characteristics for each thread according to configurable attribute objects.
- POSIX pthreads implement thread cancellation.
- POSIX pthreads enforce scheduling algorithms.
- POSIX pthreads allow for clean-up handlers for `fork(2)` calls.
- Solaris threads can be suspended and continued.
- Solaris threads implement an optimized mutex and interprocess robust mutex locks.
- Solaris threads implement daemon threads, for whose demise the process does not wait.

LinuxThreads

LinuxThreads deviates from the POSIX.1 thread specification in a number of ways, including the following:

- Calls to `getpid(2)` return a different value in each thread.
- Calls to `getppid(2)` in threads other than the main thread return the process ID of the manager thread; instead `getppid(2)` in these threads should return the same value as `getppid(2)` in the main thread.
- When one thread creates a new child process using `fork(2)`, any thread should be able to `wait(2)` on the child. However, the implementation only allows the thread that created the child to `wait(2)` on it.
- When a thread calls `execve(2)`, all other threads are terminated (as required by POSIX.1). However, the resulting process has the same PID as the thread that called `execve(2)`: it should have the same PID as the main thread.
- Threads do not share user and group IDs. This can cause complications with set-user-ID programs and can cause failures in Pthreads functions if an application changes its credentials using `seteuid(2)` or similar.
- Threads do not share a common session ID and process group ID.
- Threads do not share record locks created using `fcntl(2)`.
- The information returned by `times(2)` and `getrusage(2)` is per-thread rather than process-wide.
- Threads do not share semaphore undo values (see `semop(2)`).
- Threads do not share interval timers.
- Threads do not share a common nice value.
- POSIX.1 distinguishes the notions of signals that are directed to the process as a whole and signals are directed to individual threads. According to POSIX.1, a process-directed signal (sent using `kill(2)`, for example) should be handled by a single,

arbitrarily selected thread within the process. LinuxThreads does not support the notion of process-directed signals: signals may only be sent to specific threads.

- Threads have distinct alternate signal stack settings. However, a new thread's alternate signal stack settings are copied from the thread that created it, so that the threads initially share an alternate signal stack. (A new thread should start with no alternate signal stack defined. If two threads handle signals on their shared alternate signal stack at the same time, unpredictable program failures are likely to occur.)

Other Thread Differences

Common Solaris Threads with Linux Alternatives

Linux also uses a different naming convention for some of their threads. This is a reference list for what Solaris thread is most like what Linux thread. Some changes will need to be made for the migration.

Solaris Threads	Linux Threads	Comment
cond_broadcast	pthread_cond_broadcast	Similar invocation, requires pthread.h
cond_destroy	pthread_cond_destroy	Similar invocation, requires pthread.h
cond_init	pthread_cond_init	Similar invocation, requires pthread.h
cond_signal	pthread_cond_signal	Similar invocation, requires pthread.h
cond_timedwait	pthread_cond_timedwait	Similar invocation, requires pthread.h
cond_wait	pthread_cond_wait	Similar invocation, requires pthread.h
thr_create	pthread_create	Check for compatibility, requires pthread.h
thr_exit	pthread_exit	Check for compatibility, requires pthread.h
thr_getspecific	pthread_getspecific	Check for compatibility, requires pthread.h
thr_join	pthread_join	Check for compatibility, requires pthread.h
thr_kill	pthread_kill	Check for compatibility, requires pthread.h
thr_self	pthread_self	Check for compatibility, requires pthread.h
thr_setspecific	pthread_setspecific	Check for compatibility, requires pthread.h
rwlock_destroy	pthread_rwlock_destroy	Similar invocation, requires pthread.h
rwlock_init	pthread_rwlock_init	Check for compatibility, requires pthread.h
rw_rdlock	pthread_rwlock_rdlock	Similar invocation, requires pthread.h
rw_wrlock	pthread_rwlock_wrlock	Similar invocation, requires pthread.h
rw_unlock	pthread_rwlock_unlock	Similar invocation, requires pthread.h
rw_tryrdlock	pthread_rwlock_tryrdlock	Similar invocation, requires pthread.h
rw_trywrlock	pthread_rwlock_trywrlock	Similar invocation, requires pthread.h

Common Solaris Threads Without Linux Alternatives

There are no Linux alternatives for the following:

Solaris Threads	Linux Threads	Comment
pthread_rwlock_realtimedlock_np	None	An alternative must be found
pthread_rwlock_realtimedwrllock_np	None	An alternative must be found
thr_getprio	None	An alternative must be found
thr_keycreate	None	An alternative must be found
thr_setprio	None	An alternative must be found
thr_sigsetmask	None	An alternative must be found
thr_yield	None	An alternative must be found

Mutex Threads

These mutex threads are available in Linux. To use them, include <pthread.h>.

Solaris Mutex Threads	Linux	Comment
mutex_destroy	pthread_mutex_destroy	Similar invocation, requires pthread.h
mutex_init	pthread_mutex_init	Similar invocation, requires pthread.h
mutex_lock	pthread_mutex_lock	Similar invocation, requires pthread.h
mutex_trylock	pthread_mutex_trylock	Similar invocation, requires pthread.h
mutex_unlock	pthread_mutex_unlock	Similar invocation, requires pthread.h
pthread_mutexattr_getprioceiling	Same	Similar invocation, requires pthread.h
pthread_mutexattr_getprotocol	Same	Similar invocation, requires pthread.h
pthread_mutexattr_setprioceiling	Same	Similar invocation, requires pthread.h
pthread_mutexattr_setprotocol	Same	Similar invocation, requires pthread.h
pthread_mutex_getprioceiling	Same	Similar invocation, requires pthread.h
pthread_mutex_setprioceiling	Same	Similar invocation, requires pthread.h

The following mutex threads do not have a Linux version, alternatives will need to be found.

Solaris Mutex Threads	Linux	Comment
pthread_mutexattr_getrobust_np	None	An alternative must be found
pthread_mutexattr_setrobust_np	None	An alternative must be found
pthread_mutex_realtimedlock_np	None	An alternative must be found

Semaphores

Semaphore Concepts

A semaphore is a non-negative integer count and is generally used to co-ordinate access to resources. The initial semaphore count is set to the number of free resources, then threads slowly increment or decrement this count as resources are added or removed.

Semaphores are part of the Threads Library, but are separated here for simplicity.

Linux Compatible

Linux prefaces its semaphore routines with 'sem_' as opposed to the Solaris 'sema_'. Linux also requires the inclusion of the semaphore.h library. The invocation of these semaphores is the same, with the exception of sema_init, which has different variables.

The compatible semaphores would include:

Solaris Semaphore Call	Linux	Comment
sema_destroy	sem_destroy	Same invocation, requires #include <semaphores.h>
sema_init	sem_init	Different invocation, requires #include <semaphores.h>
sema_post	sem_post	Same invocation, requires #include <semaphores.h>
sema_trywait	sem_trywait	Same invocation, requires #include <semaphores.h>
sema_wait	sem_wait	Same invocation, requires #include <semaphores.h>

Solaris Only Semaphores

These semaphores have no Linux equivalent, as they are Solaris DDI (Device Driver Interface) specific. Alternatives must be found in these cases.

Solaris Semaphore Call	Linux	Comment
sema_p	None	An alternative must be found
sema_psig	None	An alternative must be found
sema_tryp	None	An alternative must be found
sema_v	None	An alternative must be found

Sockets

Another area of concern when porting from Solaris to Linux is in the area of sockets.

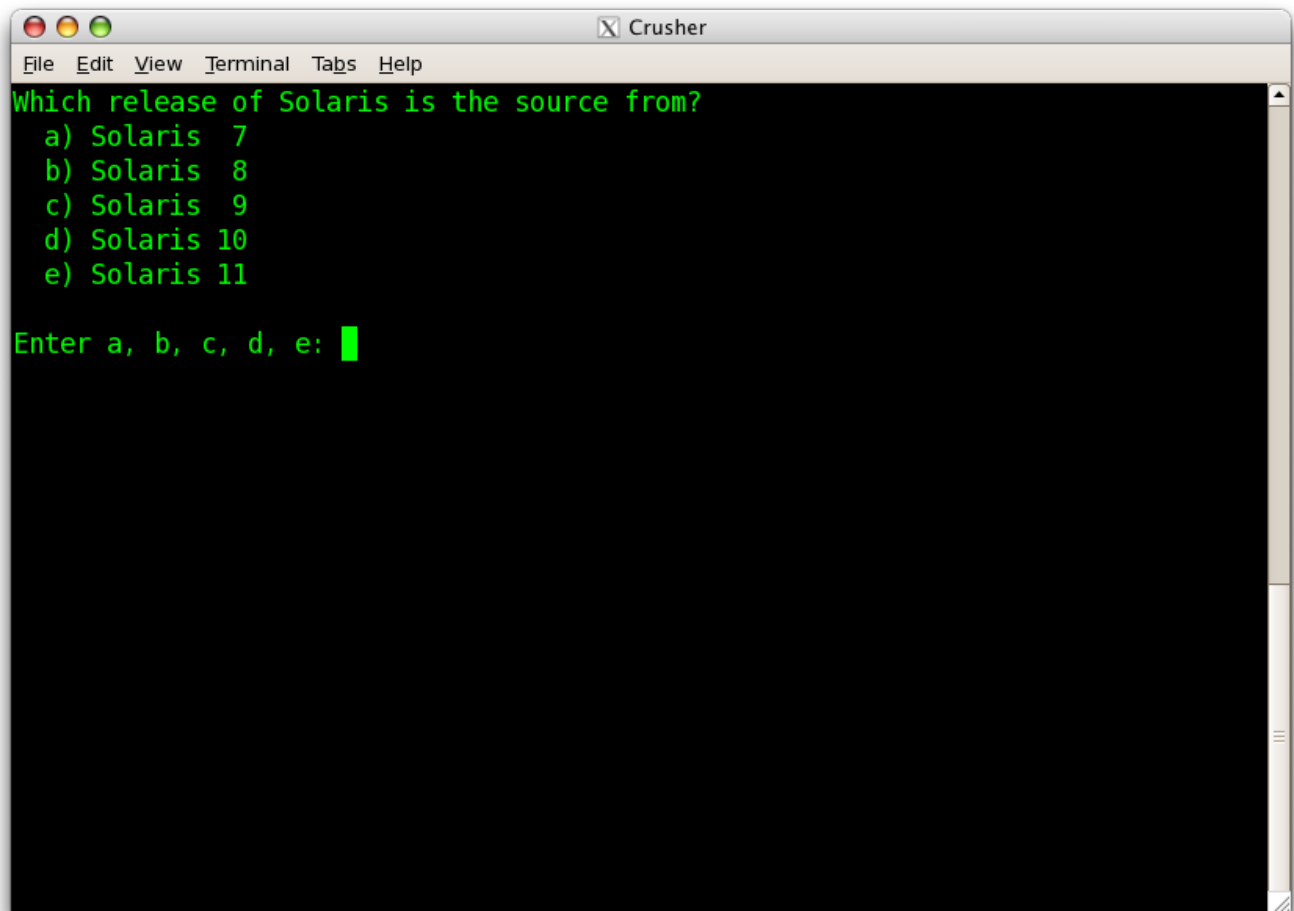
Keep in mind that the understood socket protocols differ between the two systems. Check out the socket man page from both systems to compare the defined formats and types.

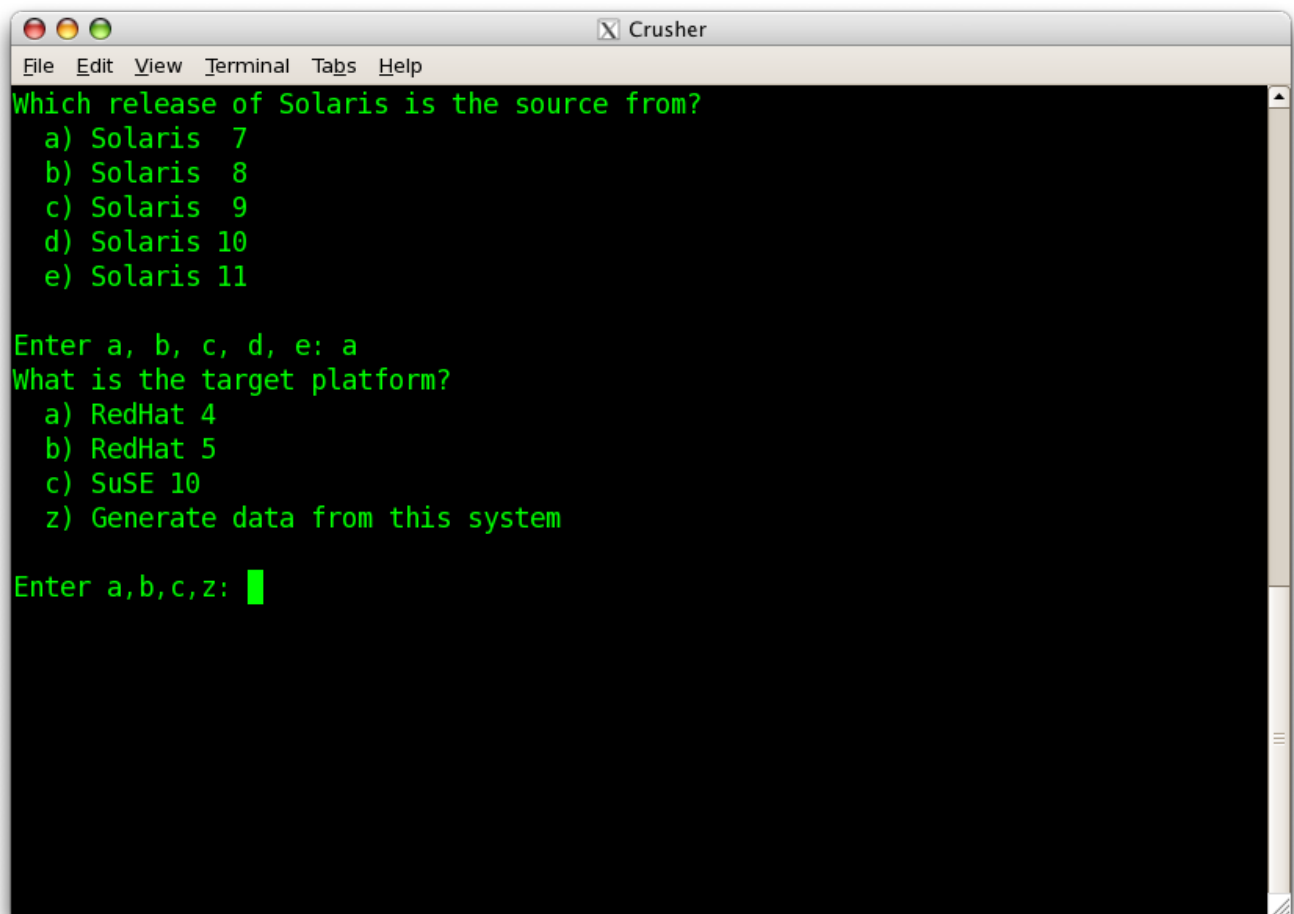
Running CountCat

As mentioned before, when CountCat is invoked the first time, it will create the database required for the code analysis.

Once you change to the root directory of the source code, run `/opt/ccat/bin/ccat`.

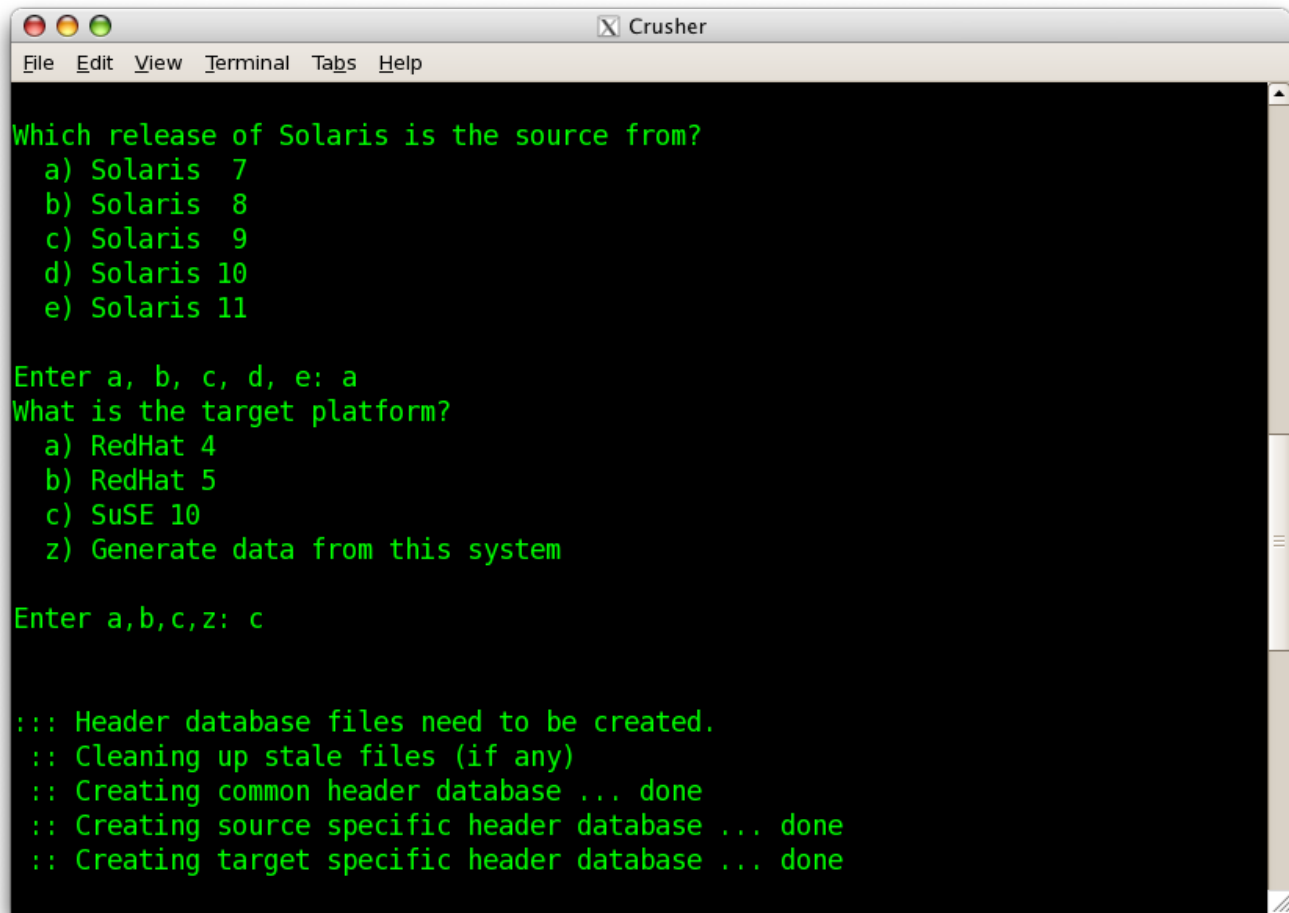
The first question asked will be 'What release of Solaris is the source from?'. Just enter the release of Solaris that the code was compiled on.





Next, the target system is determined. Currently the options are RedHat 4, RedHat 5, SUSE 10 and 'current system'. If you are performing the source migration on the server running CountCat, it is preferable to perform 'z, Generate data from this system'.

Next, with the version information, the databases will be created.



```
Crusher
File Edit View Terminal Tabs Help

Which release of Solaris is the source from?
a) Solaris 7
b) Solaris 8
c) Solaris 9
d) Solaris 10
e) Solaris 11

Enter a, b, c, d, e: a
What is the target platform?
a) RedHat 4
b) RedHat 5
c) SuSE 10
z) Generate data from this system

Enter a,b,c,z: c

::: Header database files need to be created.
:: Cleaning up stale files (if any)
:: Creating common header database ... done
:: Creating source specific header database ... done
:: Creating target specific header database ... done
```

Types of Code Audit

There are two types of code audit, a quick audit that will count the c files and header files, and the total number of lines of code. CountCat does not differentiate between lines of source code and comments.

Full Audit Example

```
Crusher
File Edit View Terminal Tabs Help

=====
F U J I T S U                               C o u n t C a t
                                           FMSccat universal version 1.3.0
Fujitsu Migration Services                Run Date: Mon Jun  1 09:50:55 EDT 2009
=====

Number of C files in the source tree:              45
Total number of lines of C code:                  35074
=====

Number of header files in the source tree:          6
Total number of lines of code in the header files: 1492
=====

Total number of C code and header files:           51
Total lines of code in above files:                36566
=====

Headers referenced by the C code:                  47
Headers included in the source code tree:           6
Common headers in Solaris and Linux:                38
Solaris only headers referenced by the source code: 3
=====

Files with System Calls requiring large modifications: 1
Files with System Calls requiring small modifications: 8
Number of files with Socket calls in code:          1
Union declarations in headers:                      1
=====

Script Hunting:
sh scripts   :    1 | total lines :    512
bash scripts :    0 | total lines :      0
ksh scripts  :    0 | total lines :      0
csh scripts  :    0 | total lines :      0
tcsh scripts :    0 | total lines :      0
perl scripts :    0 | total lines :      0

Script information in script_detail

-----
Please see ccat.html and ccat.csv for more detail  Copyright Fujitsu 2008-2009
-----
[azp80@crusher mailx]$
```

CountCat output Files

ccat.html

ccat.html has all the information from the console output, and also lists the files found, counts and database matches in a summary manner.

More ccat.html detail.

FUJITSU THE POSSIBILITIES ARE INFINITE

Solaris to Linux C Source Analyser

ccat version: 1.3.0
Report Run: Mon Jun 1 09:50:55 EDT 2009

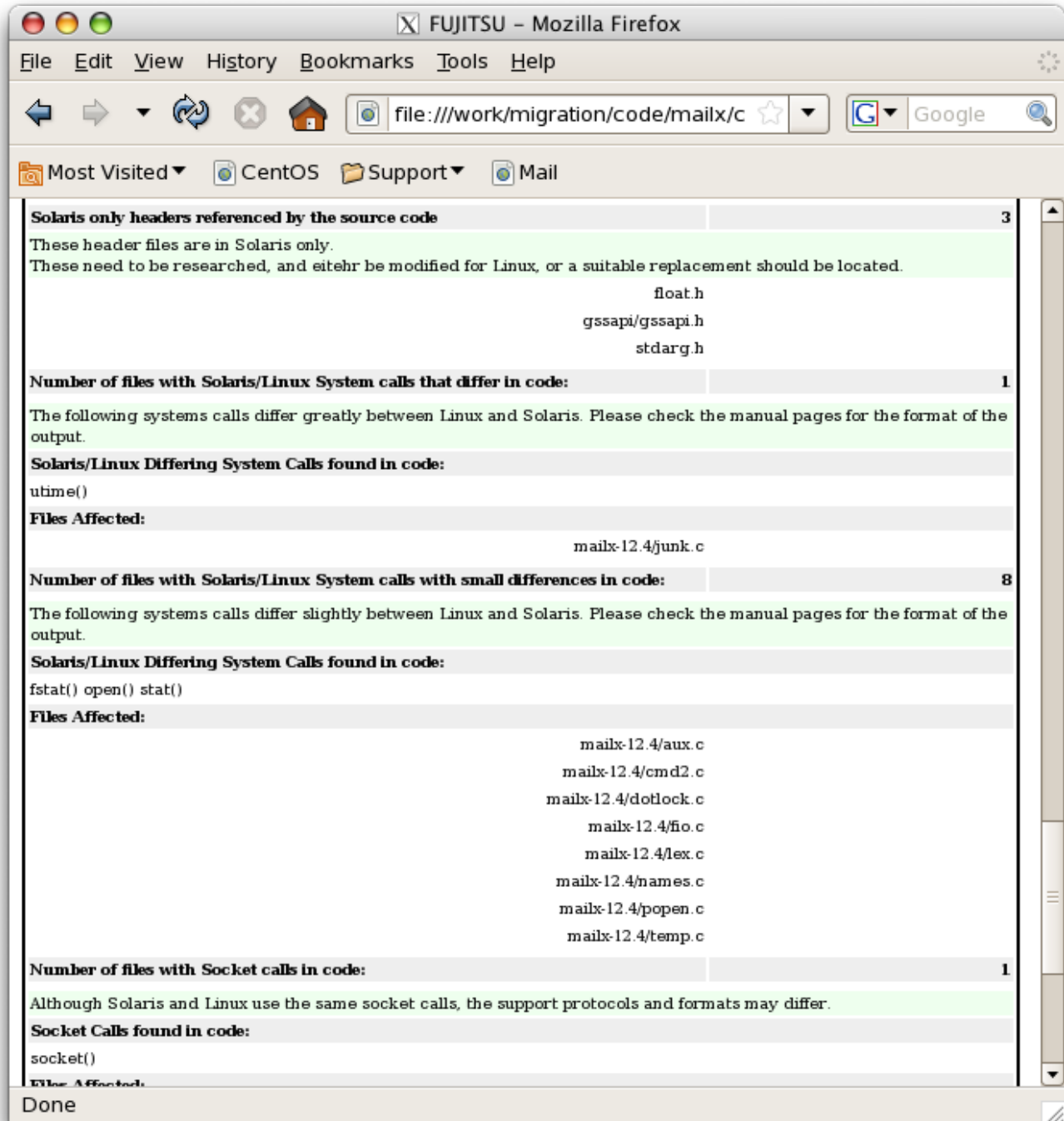
System Information

Arch: i686
Operating System: Linux 2.6.18-128.1.10.el5
Source: /home/azp80/migration/code/mailx

C Files

main.c	565
vars.c	287
hmac.c	113
openssl.c	1152
mime.c	1623
macro.c	387
cache.c	814
md5.c	329
dotlock.c	278
edit.c	253
ssl.c	399
sendout.c	1532
v7.local.c	107
imap_search.c	742
junk.c	1205
imap_gssapi.c	286
lzw.c	698
getopt.c	158
cmd1.c	1145
thread.c	748

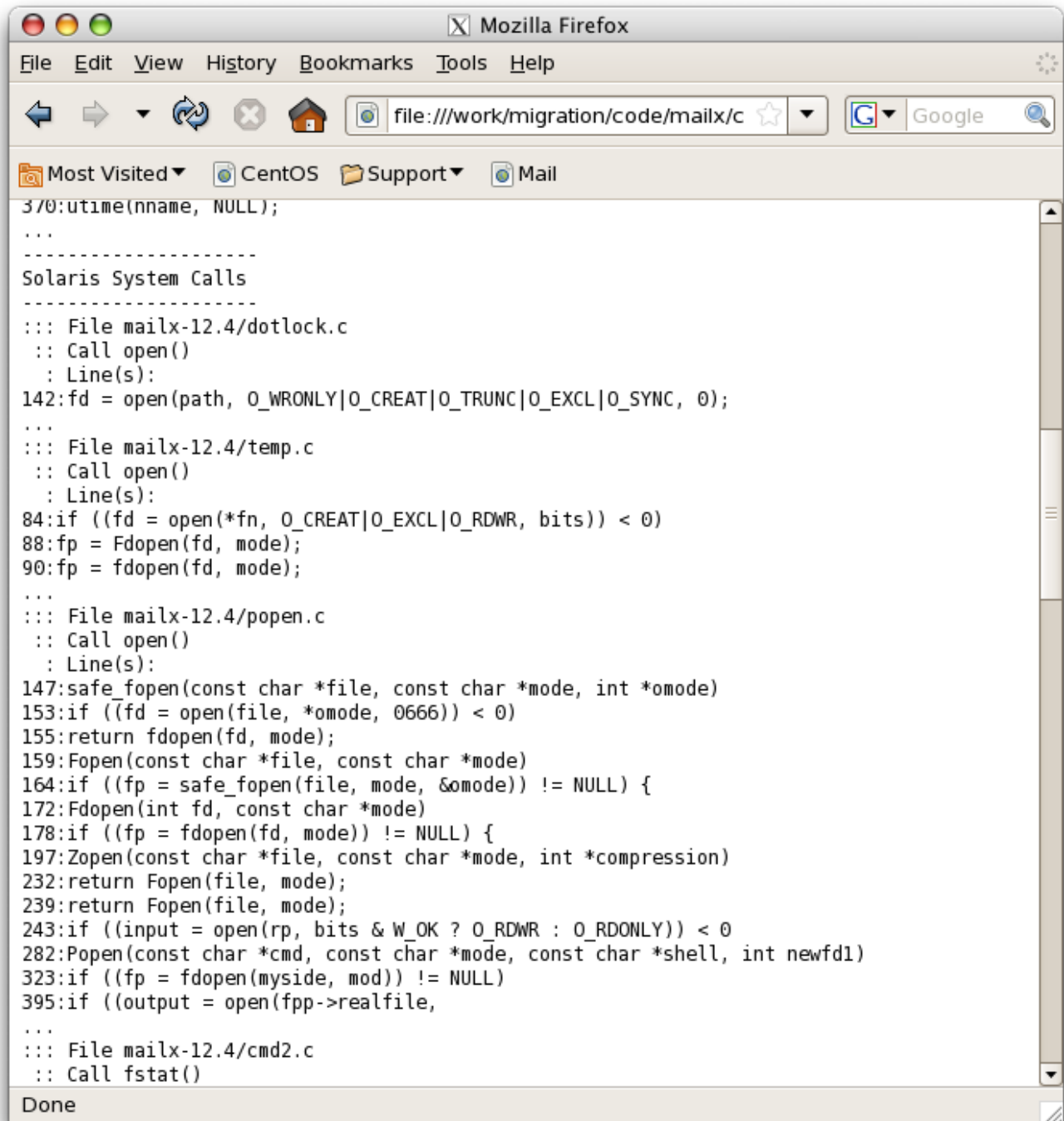
Done



At the bottom, there is a link to [ccat_detail.html](#)

ccat_detail.html

There is a much more detailed report that is generated at the same time. This file will display the filename, the match that was detected, and the line of code and the line number.

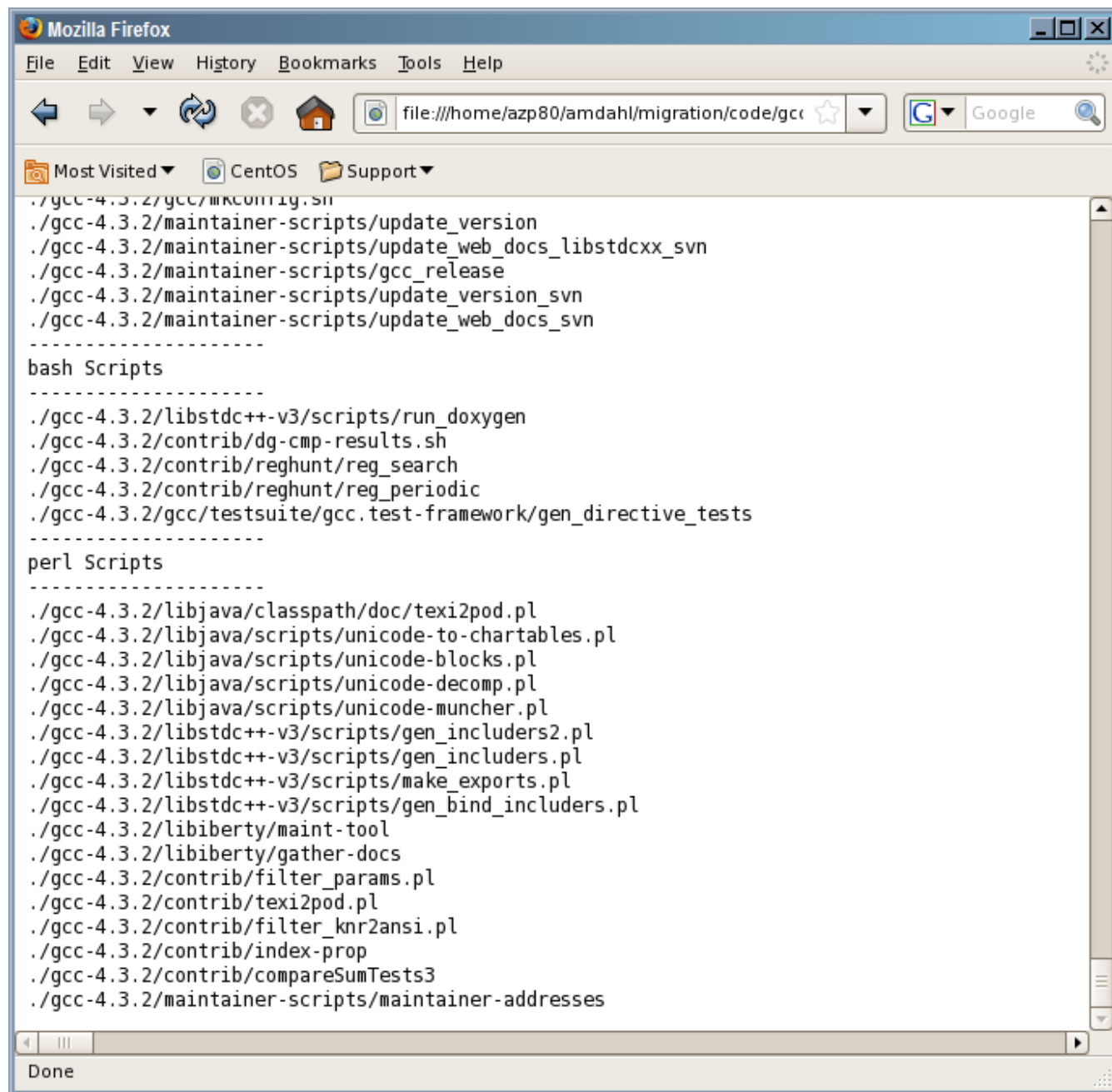


The screenshot shows a Mozilla Firefox browser window. The title bar says "Mozilla Firefox". The menu bar includes "File", "Edit", "View", "History", "Bookmarks", "Tools", and "Help". The address bar shows the file path "file:///work/migration/code/mailx/c". Below the address bar, there are "Most Visited" links for "CentOS", "Support", and "Mail". The main content area displays a text file with the following content:

```
370:utime(nname, NULL);
...
-----
Solaris System Calls
-----
::: File mailx-12.4/dotlock.c
:: Call open()
: Line(s):
142:fd = open(path, O_WRONLY|O_CREAT|O_TRUNC|O_EXCL|O_SYNC, 0);
...
::: File mailx-12.4/temp.c
:: Call open()
: Line(s):
84:if ((fd = open(*fn, O_CREAT|O_EXCL|O_RDWR, bits)) < 0)
88:fp = fdopen(fd, mode);
90:fp = fdopen(fd, mode);
...
::: File mailx-12.4/popen.c
:: Call open()
: Line(s):
147:safe_fopen(const char *file, const char *mode, int *omode)
153:if ((fd = open(file, *omode, 0666)) < 0)
155:return fdopen(fd, mode);
159:Fopen(const char *file, const char *mode)
164:if ((fp = safe_fopen(file, mode, &omode)) != NULL) {
172:Fdopen(int fd, const char *mode)
178:if ((fp = fdopen(fd, mode)) != NULL) {
197:Zopen(const char *file, const char *mode, int *compression)
232:return Fopen(file, mode);
239:return Fopen(file, mode);
243:if ((input = open(rp, bits & W_OK ? O_RDWR : O_RDONLY)) < 0)
282:Popen(const char *cmd, const char *mode, const char *shell, int newfd1)
323:if ((fp = fdopen(myside, mod)) != NULL)
395:if ((output = open(fpp->realfile,
...
::: File mailx-12.4/cmd2.c
:: Call fstat()
```

The status bar at the bottom of the browser window says "Done".

Here is an example of the output from the script hunting:



```
./gcc-4.3.2/gcc/mkconfig.sh
./gcc-4.3.2/maintainer-scripts/update_version
./gcc-4.3.2/maintainer-scripts/update_web_docs_libstdcxx_svn
./gcc-4.3.2/maintainer-scripts/gcc_release
./gcc-4.3.2/maintainer-scripts/update_version_svn
./gcc-4.3.2/maintainer-scripts/update_web_docs_svn
-----
bash Scripts
-----
./gcc-4.3.2/libstdc++-v3/scripts/run_doxygen
./gcc-4.3.2/contrib/dg-cmp-results.sh
./gcc-4.3.2/contrib/reghunt/reg_search
./gcc-4.3.2/contrib/reghunt/reg_periodic
./gcc-4.3.2/gcc/testsuite/gcc.test-framework/gen_directive_tests
-----
perl Scripts
-----
./gcc-4.3.2/libjava/classpath/doc/texi2pod.pl
./gcc-4.3.2/libjava/scripts/unicode-to-chartables.pl
./gcc-4.3.2/libjava/scripts/unicode-blocks.pl
./gcc-4.3.2/libjava/scripts/unicode-decomp.pl
./gcc-4.3.2/libjava/scripts/unicode-muncher.pl
./gcc-4.3.2/libstdc++-v3/scripts/gen_includers2.pl
./gcc-4.3.2/libstdc++-v3/scripts/gen_includers.pl
./gcc-4.3.2/libstdc++-v3/scripts/make_exports.pl
./gcc-4.3.2/libstdc++-v3/scripts/gen_bind_includers.pl
./gcc-4.3.2/libiberty/maint-tool
./gcc-4.3.2/libiberty/gather-docs
./gcc-4.3.2/contrib/filter_params.pl
./gcc-4.3.2/contrib/texi2pod.pl
./gcc-4.3.2/contrib/filter_knr2ansi.pl
./gcc-4.3.2/contrib/index-prop
./gcc-4.3.2/contrib/compareSumTests3
./gcc-4.3.2/maintainer-scripts/maintainer-addresses
```

A csv formatted file is created, which can be brought into a spread sheet.