Debugging & Analyzing Bugs using PIN tool

MS Project submitted by

Shaik Inzamam Ul Haq	**Jonathan Raj**	**Sai Manideep Reddy**
Computer Science	*Computer Science*	*Computer Science*
SUNY Polytechnic	*SUNY Polytechnic*	*SUNY Polytechnic*
Institute Utica, NY	*Institute* Utica, NY	*Institute* Utica, NY
U00348563	U00348780	U00349236
shaiks3@sunypoly.edu	**katikaj@sunypoly.edu**	**lakkirs@sunypoly.edu**

Supervised by

Dr. Tarannum Shaila Zaman

Date: 12/12/2024

Master of Science in Computer and Information Sciences

Department of Computer Science

State University of New York Polytechnic Institute

Debugging and Analyzing Bugs using PIN tool, a project by Inzamam Ul Haq Shaik (U00348563); Jonathan Raj Katikala (U00348780); Sai Manideep Reddy Lakkireddy (U00349236), is approved and recommended for acceptance as a final project in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences, SUNY Polytechnic Institute

Date:

Tarannum Shaila Zaman
Associate Professor, Computer Science

Chen-Fu Chiang
Associate Professor, Computer Science

William Confer
Associate Professor, Computer Science

STUDENT DECLARATION

We declare that this project is our own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Inzamam Ul Haq Shaik (U00348563)

Jonathan Raj Katikala (U00348780)

Sai Manideep Reddy L (U00349236)

ACKNOWLEDGE

We would like to express our extreme gratitude to Prof. Tarannum Zaman who has been a great help during the project and academic year. Her guidance, support and motivation were key factors in this project. Her comments and suggestions were very helpful during the project. Likewise, we would like to express our gratitude towards the graduate faculty of the Department of Computer Information Science. They have always supported the students in learning and developing new skills by providing us with opportunities to work on various projects and assignments during the academic year. Their encouragement and support kept us ambitious.

ABSTRACT

System Call Tracing of Linux Coreutils

System calls are the fundamental interface between user-level applications and the operating system kernel. Understanding their behavior is crucial for debugging, performance optimization, and security analysis. This project focuses on tracing system calls made by Linux coreutils, a suite of essential command-line utilities.

By employing Intel's PIN Tool, we developed a custom tool to intercept and log system calls during the execution of coreutils binaries. The resulting strace.out files provide detailed information about each system call, including its name, arguments, and return value. This data enables in-depth analysis of coreutils' internal workings, aiding in bug reproduction, performance profiling, and security vulnerability assessment.

Furthermore, we explored the challenges of managing multiple coreutils versions for historical bug reproduction. By addressing dependency and compatibility issues, we established a robust environment for analyzing past system behavior.

This research contributes to the broader understanding of system call behavior and provides valuable insights for software developers, security researchers, and system administrators.

Table of Content

INTRODUCTION

System calls provide a fundamental connection between user applications and the operating system kernel, allowing critical tasks such as file operations, process management, and memory handling. A deep understanding of how system call's function is important for effective debugging, analyzing performance, and replicating software issues.

This project explores system call tracing with a focus on Linux Coreutils, a crucial collection of command-line tools. Utilizing Intel's PinTool, we carefully instrumented Coreutils to generate detailed system call traces, allowing for an in-depth examination of their internal behavior. The recorded traces, saved in strace.out files, offer valuable insights for debugging, performance tuning, and analyzing historical bugs.

Furthermore, this project involved the meticulous installation and management of multiple coreutils versions to facilitate the reproduction of known bugs. We encountered and successfully addressed various challenges, including dependency mismatches and compatibility issues, by employing systematic debugging techniques and environment isolation strategies.

This document highlights our detailed approach, key findings, and valuable insights from utilizing PinTool to trace and study system calls. By thoroughly examining the behavior of system calls, we aim to drive progress in areas such as software development, debugging techniques, and security evaluation.

APPROACH

1. Analysis of technique

Static Analysis

- **Advantages:** We can identify the issues early in the development.
- **Limitations:** May produce false positives, or we might skip subtle bugs, especially those that are clearly under specific execution conditions.

Dynamic Analysis

- **Advantages:** We can detect runtime errors and performance bottlenecks.
- **Limitations:** It may require extensive testing and may not cover all possible execution paths.

In-Depth Bug Analysis

In-Depth Bug Analysis that Leverages Pin Tool in this approach, which utilizes the Pin tool for in-depth bug analysis, offers several advantages over existing techniques:

- **Precise Instrumentation:** Pin allows for the right process to produce binary code, enabling the collection of detailed execution traces at the instruction level.
- **Flexibility:** Pin's architecture enables the creation of custom analysis tools (Pin tools) tailored to specific bug analysis tasks.
- **Scalability:** Pin can handle large and complex programs, making it suitable for analyzing real-world software systems.
- **Complementary to Existing Techniques:** Our approach can be used with static and dynamic analysis to provide a more comprehensive understanding of software bugs.

By harnessing the capabilities of Pin alongside our pioneering analysis methodologies, we present a groundbreaking framework for comprehensive bug diagnosis. Our key aims are:

- **Decoding Underlying Causes**: Precisely trace the origins of intricate bugs, including memory anomalies, synchronization issues, and algorithmic flaws, ensuring a deep understanding of their nature.
- **Streamlining Bug Resolution**: Offer actionable intelligence that enables developers to rectify issues swiftly and with greater precision, reducing both time and effort.
- **Fortifying Software Integrity**: Foster the creation of dependable, secure, and high-performing software systems by addressing vulnerabilities at their core.

By combining the power of Pin with our proposed analysis techniques, which is in-depth bug analysis, where we aim to:

Identify Root Causes: Accurately pinpoint the underlying causes of bugs, including memory corruption, race conditions, and incorrect algorithm implementation.

Improve Bug Fixing: Provide insights to developers to facilitate efficient bug fixing.

Enhance Software Quality: Contribute to the development of more reliable and secure software systems.

2. Installing and Configuring Coreutils Versions

To reproduce historical bugs in Linux core utilities (coreutils), we installed multiple versions of coreutils and addressed various challenges during the installation process.

2.1. Installed Coreutils Versions

The following versions were installed to analyze differences and reproduce specific bugs:

Reproduced Versions of CoreUtils
Coreutils - 5.2.0
Coreutils - 5.2.1
Coreutils - 5.94
Coreutils - 6.9
Coreutils - 8.2
Coreutils - 8.4

2.2. Installation Procedure

Each version of coreutils was downloaded as a tarball from GNU archives. The installation steps included:

Extracting the tarball:	`tar -xvzf coreutils-<version>.tar.gz` `cd coreutils-<version>`

Configuring and building the source:	`./configure`
	`make`
	`sudo make install`

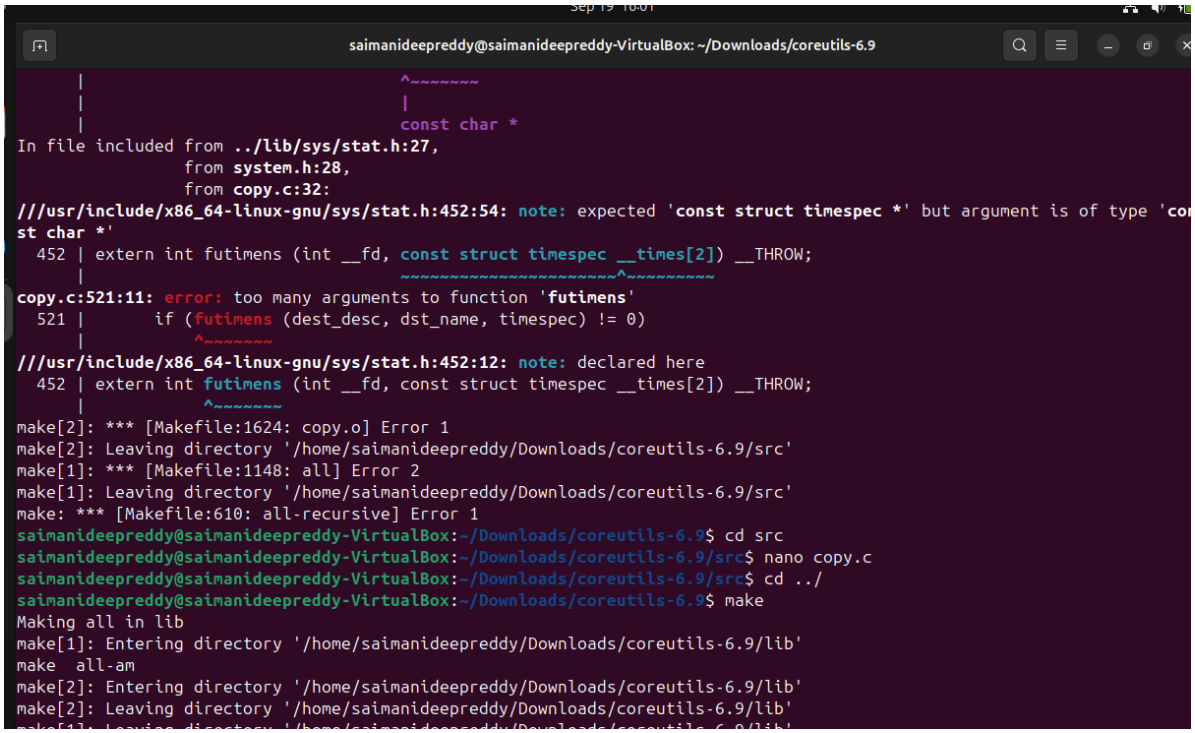
Verifying the installation:	`mv –version`

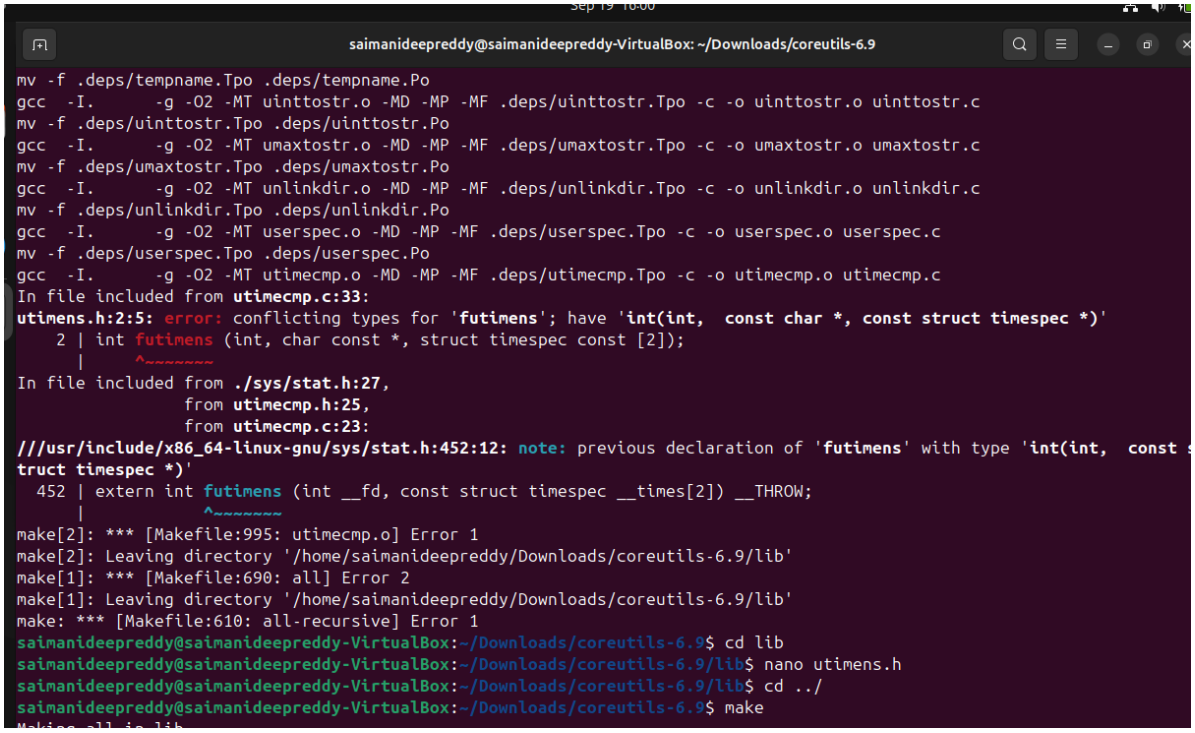
INSTALLATION ERRORS

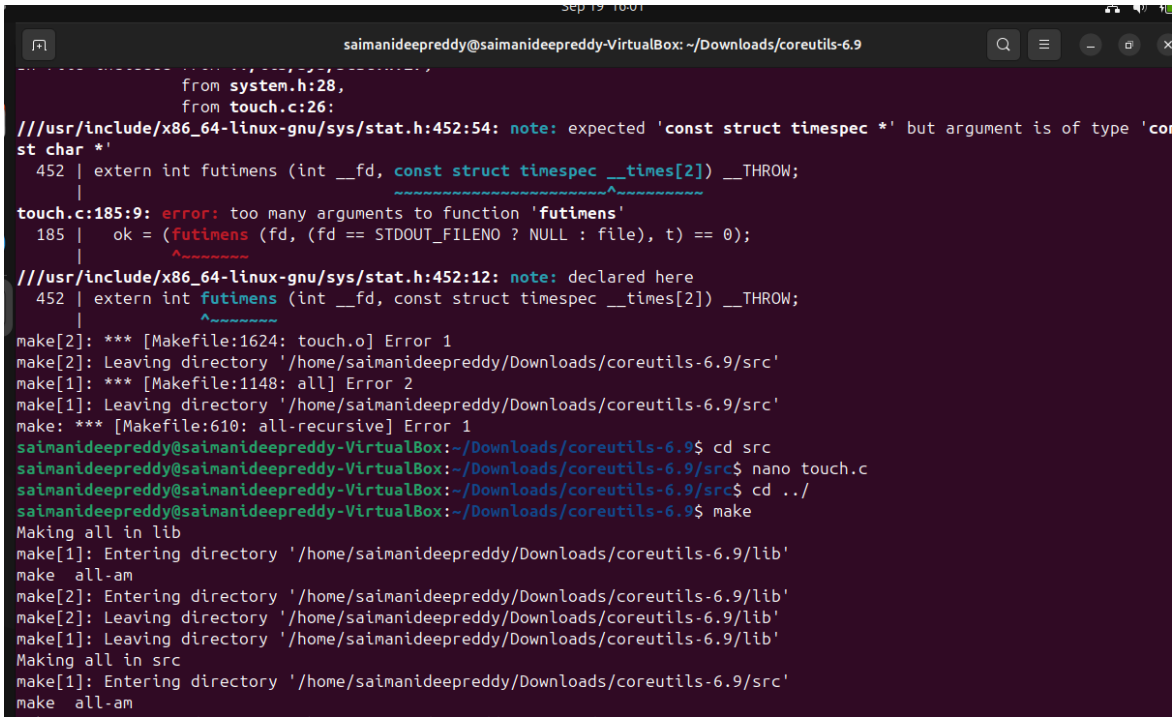
Challenges in Coreutils version 6.9

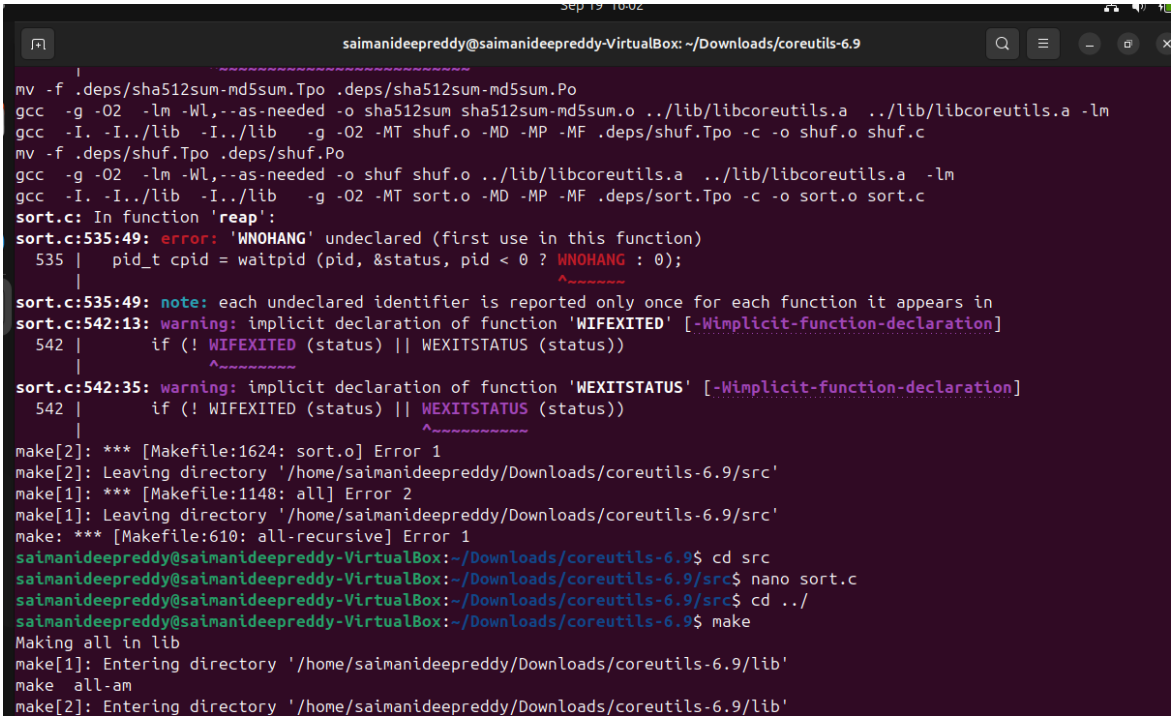
The futimens function and its associated header were included in stat.h, copy.c and touch.c using conditional compilation. A macro was defined with #define to control the inclusion, and #ifdef was applied to ensure futimens is only compiled when the macro is defined. This approach optimizes compatibility and code efficiency across different coreutils.

In the sort.c file, To resolve the errors related to the WNOHANG, WIFEXITED, and WEXITSTATUS functions by adding the necessary headers and defining macros. The functions were prefixed with my_ to create custom implementations. This approach ensures proper function declarations and eliminates implicit definition warnings.



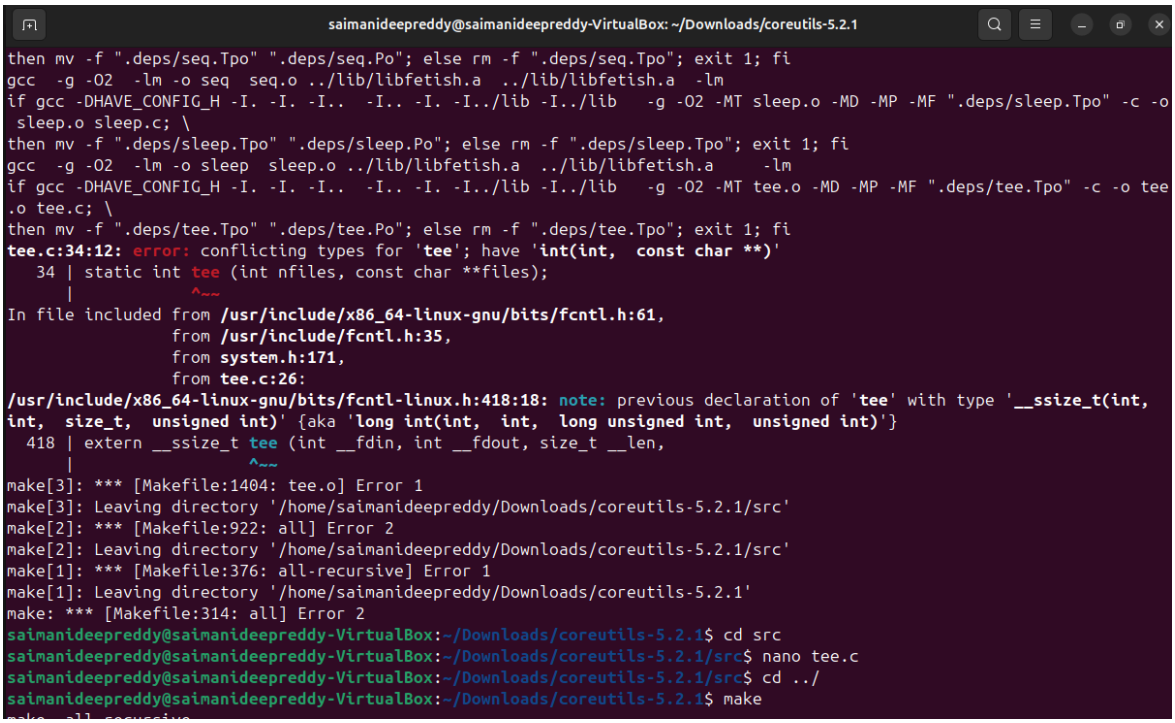

saimanideepreddy@saimanideepreddy-VirtualBox: ~/Downloads/coreutils-6.9
mv -f .deps/tempname.Tpo .deps/tempname.Po
gcc -I. -g -O2 -MT uinttostr.o -MD -MP -MF .deps/uinttostr.Tpo -c -o uinttostr.o uinttostr.c
mv -f .deps/uinttostr.Tpo .deps/uinttostr.Po
gcc -I. -g -O2 -MT umaxtostr.o -MD -MP -MF .deps/umaxtostr.Tpo -c -o umaxtostr.o umaxtostr.c
mv -f .deps/umaxtostr.Tpo .deps/umaxtostr.Po
gcc -I. -g -O2 -MT unlinkdir.o -MD -MP -MF .deps/unlinkdir.Tpo -c -o unlinkdir.o unlinkdir.c
mv -f .deps/unlinkdir.Tpo .deps/unlinkdir.Po
gcc -I. -g -O2 -MT userspec.o -MD -MP -MF .deps/userspec.Tpo -c -o userspec.o userspec.c
mv -f .deps/userspec.Tpo .deps/userspec.Po
gcc -I. -g -O2 -MT utimecmp.o -MD -MP -MF .deps/utimecmp.Tpo -c -o utimecmp.o utimecmp.c
In file included from utimecmp.c:33:
utimens.h:2:5: error: conflicting types for 'futimens'; have 'int(int, const char *, const struct timespec *)'
2 | int futimens (int, char const *, struct timespec const [2]);
| ^~~~~~~~
In file included from ./sys/stat.h:27,
from utimecmp.h:25,
from utimecmp.c:23:
///usr/include/x86_64-linux-gnu/sys/stat.h:452:12: note: previous declaration of 'futimens' with type 'int(int, const s
truct timespec *)'
452 | extern int futimens (int __fd, const struct timespec __times[2]) __THROW;
| ^~~~~~~~
make[2]: *** [Makefile:995: utimecmp.o] Error 1
make[2]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-6.9/lib'
make[1]: *** [Makefile:690: all] Error 2
make[1]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-6.9/lib'
make: *** [Makefile:610: all-recursive] Error 1
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-6.9$ cd lib
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-6.9/lib$ nano utimens.h
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-6.9/lib$ cd ../
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-6.9$ make


saimanideepreddy@saimanideepreddy-VirtualBox: ~/Downloads/coreutils-6.9
from system.h:28,
from touch.c:26:
///usr/include/x86_64-linux-gnu/sys/stat.h:452:54: note: expected 'const struct timespec *' but argument is of type 'co
nst char *'
452 | extern int futimens (int __fd, const struct timespec __times[2]) __THROW;
| ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~
touch.c:185:9: error: too many arguments to function 'futimens'
185 | ok = (futimens (fd, (fd == STDOUT_FILENO ? NULL : file), t) == 0);
| ^~~~~~~~
///usr/include/x86_64-linux-gnu/sys/stat.h:452:12: note: declared here
452 | extern int futimens (int __fd, const struct timespec __times[2]) __THROW;
| ^~~~~~~~
make[2]: *** [Makefile:1624: touch.o] Error 1
make[2]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-6.9/src'
make[1]: *** [Makefile:1148: all] Error 2
make[1]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-6.9/src'
make: *** [Makefile:610: all-recursive] Error 1
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-6.9$ cd src
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-6.9/src$ nano touch.c
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-6.9/src$ cd ../
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-6.9$ make
Making all in lib
make[1]: Entering directory '/home/saimanideepreddy/Downloads/coreutils-6.9/lib'
make all-am
make[2]: Entering directory '/home/saimanideepreddy/Downloads/coreutils-6.9/lib'
make[2]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-6.9/lib'
make[1]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-6.9/lib'
Making all in src
make[1]: Entering directory '/home/saimanideepreddy/Downloads/coreutils-6.9/src'
make all-am



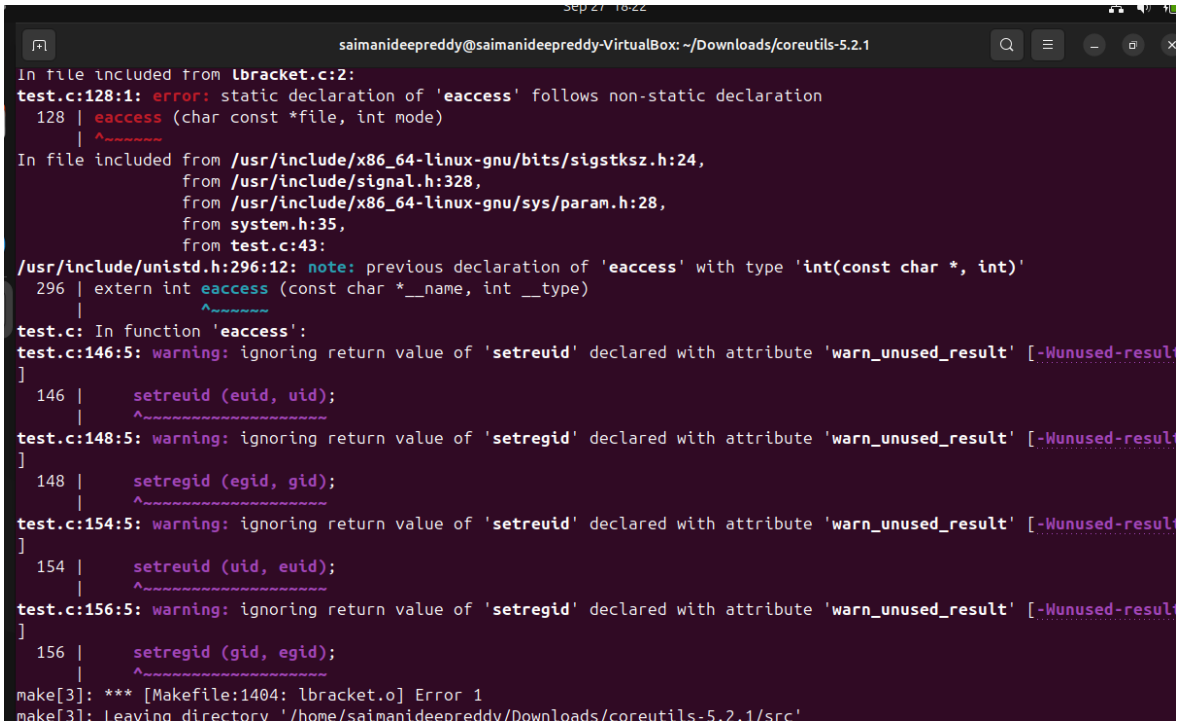
Challenges in Coreutils version 5.2.1

In tee.c, the error related to conflicting types for tee was resolved by renaming the custom tee function to my_tee. This eliminated the conflict with the system tee function and ensured consistent function declarations.



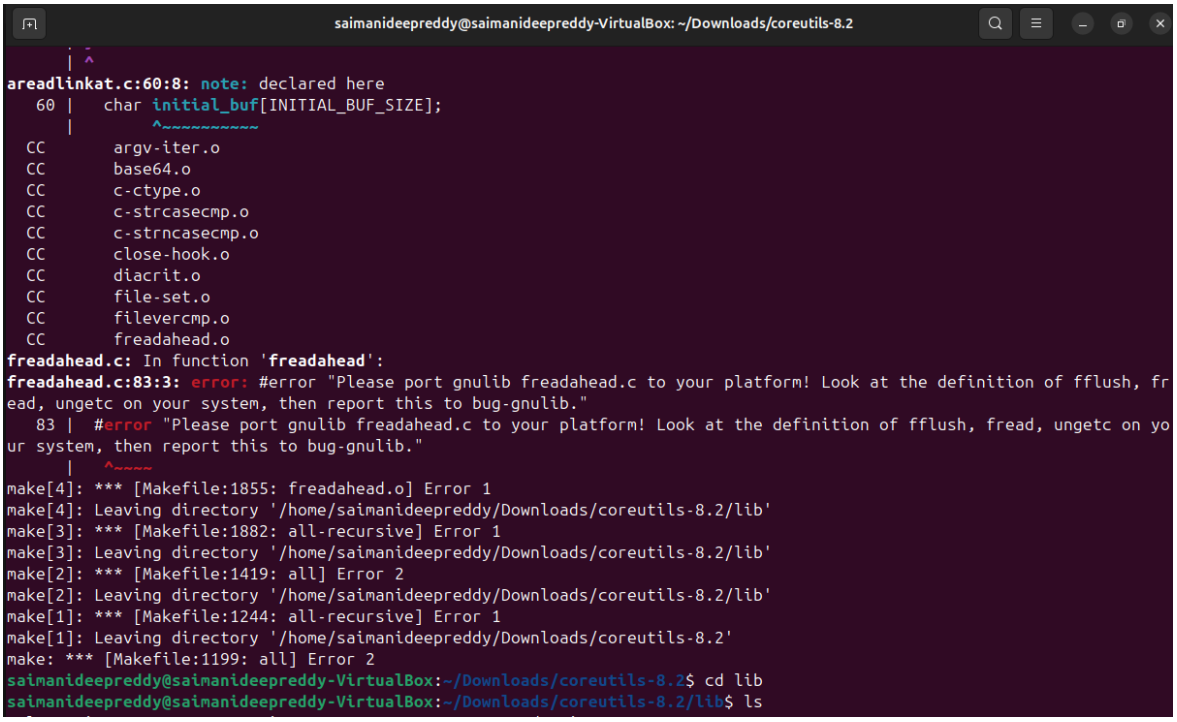
In test.c, the error related to the static declaration of eaccess following a non-static declaration was resolved by renaming the function to my_eaccess. This ensured proper scoping and eliminated conflicts between the static and non-static declarations.

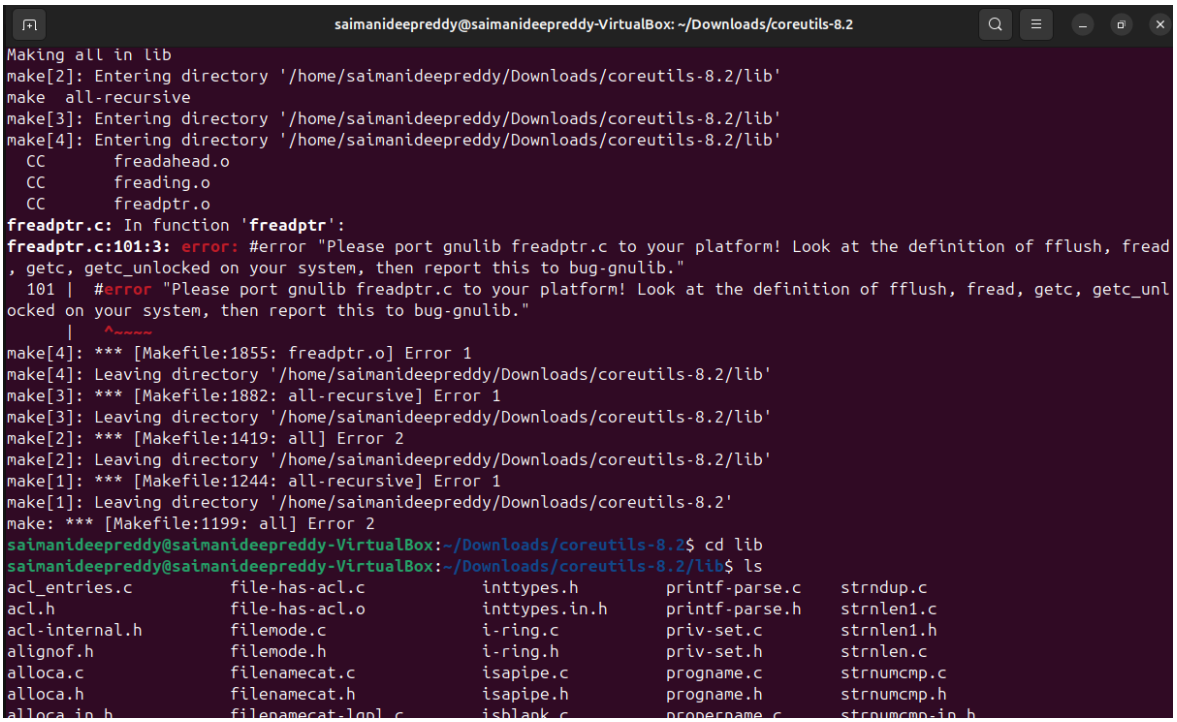
To resolve the warning in test.c regarding the ignored return value of setregid (marked with the warn_unused_result attribute), the return value was explicitly captured and checked. Appropriate error handling was implemented to ensure that any failure of setregid is properly addressed, enhancing the robustness of the code.

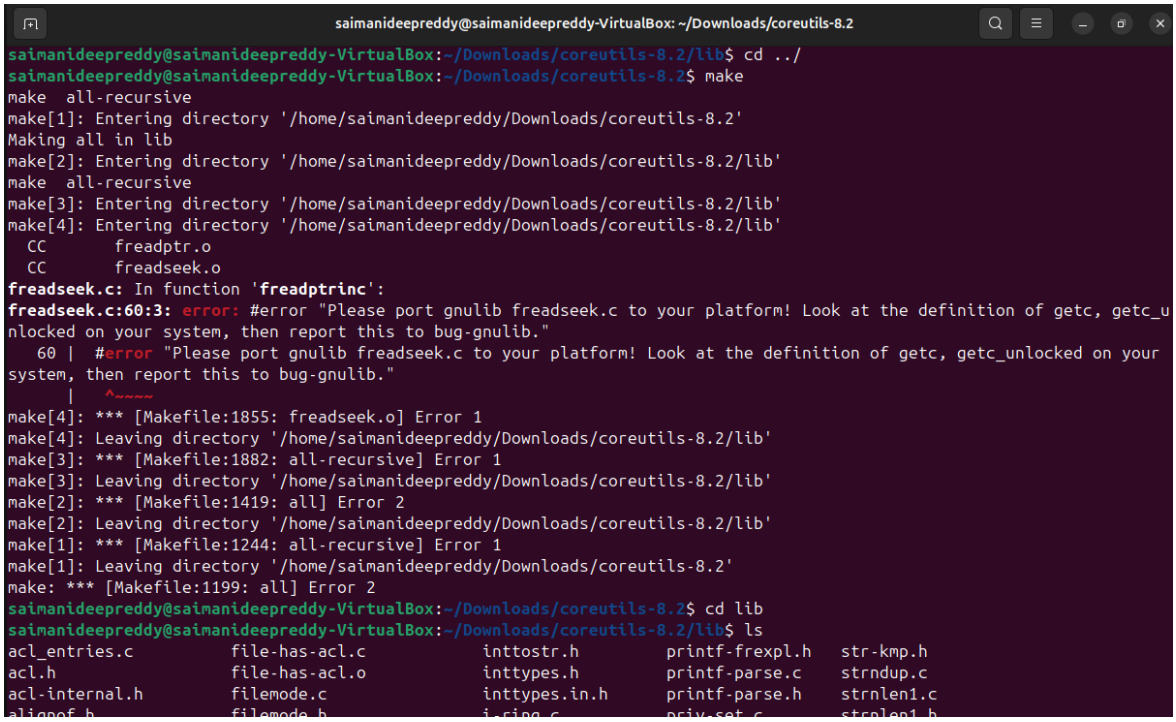


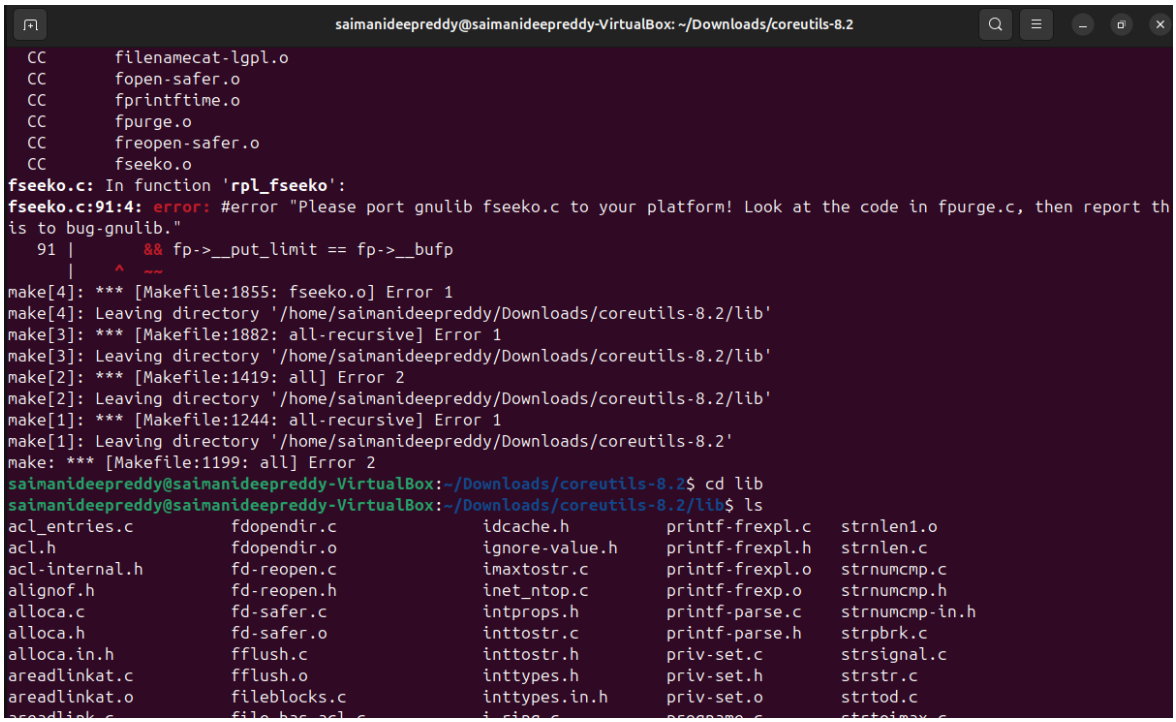
Challenges in Coreutils version 8.2

In freadahead.c, freadptr.c, freadseek.c, and fseterr.c, the error message "undeclared identifier" signals that a necessary function or variable is not recognized. This problem often stems from missing declarations or headers, which can lead to challenges during development. It emphasizes the fact that developers do not always have full control over their development environment, making it essential to include well-documented comments and precise declarations to enhance code readability and maintainability. Similar issues have been observed in versions 8.2 and 8.4 of Coreutils.


saimanideepreddy@saimanideepreddy-VirtualBox: ~/Downloads/coreutils-8.2
| ^
areadlinkat.c:60:8: note: declared here
60 | char initial_buf[INITIAL_BUF_SIZE];
| ^~~~~~~~~~~
CC argv-iter.o
CC base64.o
CC c-ctype.o
CC c-strcasecmp.o
CC c-strncasecmp.o
CC close-hook.o
CC diacrit.o
CC file-set.o
CC filevercmp.o
CC freadahead.o
freadahead.c: In function 'freadahead':
freadahead.c:83:3: error: #error "Please port gnulib freadahead.c to your platform! Look at the definition of fflush, fread, ungetc on your system, then report this to bug-gnulib."
83 | #error "Please port gnulib freadahead.c to your platform! Look at the definition of fflush, fread, ungetc on your system, then report this to bug-gnulib."
| ^~~~~
make[4]: *** [Makefile:1855: freadahead.o] Error 1
make[4]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[3]: *** [Makefile:1882: all-recursive] Error 1
make[3]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[2]: *** [Makefile:1419: all] Error 2
make[2]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[1]: *** [Makefile:1244: all-recursive] Error 1
make[1]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2'
make: *** [Makefile:1199: all] Error 2
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-8.2$ cd lib
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-8.2/lib$ ls

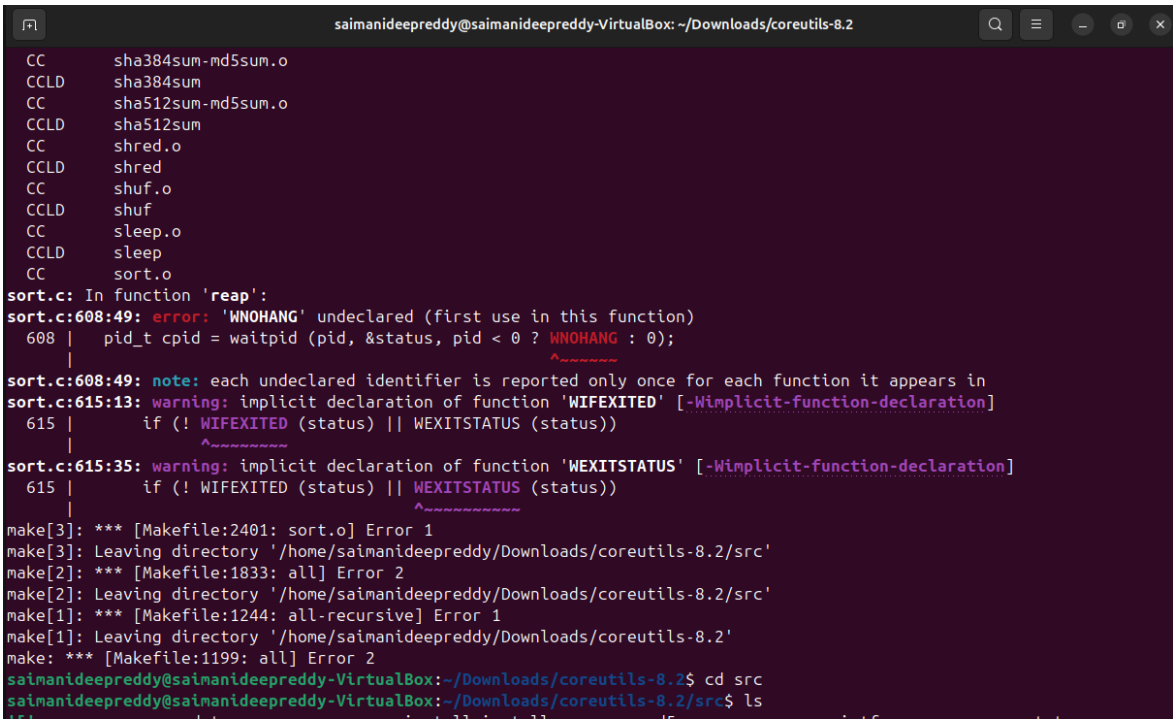

saimanideepreddy@saimanideepreddy-VirtualBox: ~/Downloads/coreutils-8.2
Making all in lib
make[2]: Entering directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make all-recursive
make[3]: Entering directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[4]: Entering directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
CC freadahead.o
CC freading.o
CC freadptr.o
freadptr.c: In function 'freadptr':
freadptr.c:101:3: error: #error "Please port gnulib freadptr.c to your platform! Look at the definition of fflush, fread, getc, getc_unlocked on your system, then report this to bug-gnulib."
101 | #error "Please port gnulib freadptr.c to your platform! Look at the definition of fflush, fread, getc, getc_unlocked on your system, then report this to bug-gnulib."
| ^~~~~
make[4]: *** [Makefile:1855: freadptr.o] Error 1
make[4]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[3]: *** [Makefile:1882: all-recursive] Error 1
make[3]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[2]: *** [Makefile:1419: all] Error 2
make[2]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[1]: *** [Makefile:1244: all-recursive] Error 1
make[1]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2'
make: *** [Makefile:1199: all] Error 2
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-8.2$ cd lib
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-8.2/lib$ ls
acl_entries.c file-has-acl.c inttypes.h printf-parse.c strndup.c
acl.h file-has-acl.o inttypes.in.h printf-parse.h strnlen1.c
acl-internal.h filemode.c i-ring.c priv-set.c strnlen1.h
alignof.h filemode.h i-ring.h priv-set.h strnlen.c
alloca.c filenamecat.c isapipe.c progname.c strnumcmp.c
alloca.h filenamecat.h isapipe.h progname.h strnumcmp.h


saimanideepreddy@saimanideepreddy-VirtualBox: ~/Downloads/coreutils-8.2
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-8.2/lib$ cd ../
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-8.2$ make
make all-recursive
make[1]: Entering directory '/home/saimanideepreddy/Downloads/coreutils-8.2'
Making all in lib
make[2]: Entering directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make all-recursive
make[3]: Entering directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[4]: Entering directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
  CC   freadptr.o
  CC   freadseek.o
freadseek.c: In function 'freadptrinc':
freadseek.c:60:3: error: #error "Please port gnulib freadseek.c to your platform! Look at the definition of getc, getc_unlocked on your system, then report this to bug-gnulib."
   60 | #error "Please port gnulib freadseek.c to your platform! Look at the definition of getc, getc_unlocked on your system, then report this to bug-gnulib."
      | ^~~~~
make[4]: *** [Makefile:1855: freadseek.o] Error 1
make[4]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[3]: *** [Makefile:1882: all-recursive] Error 1
make[3]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[2]: *** [Makefile:1419: all] Error 2
make[2]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[1]: *** [Makefile:1244: all-recursive] Error 1
make[1]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2'
make: *** [Makefile:1199: all] Error 2
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-8.2$ cd lib
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-8.2/lib$ ls
acl_entries.c file-has-acl.c inttostr.h printf-frexpl.h str-kmp.h
acl.h file-has-acl.o inttypes.h printf-parse.c strndup.c
acl-internal.h filemode.c inttypes.in.h printf-parse.h strnlen1.c

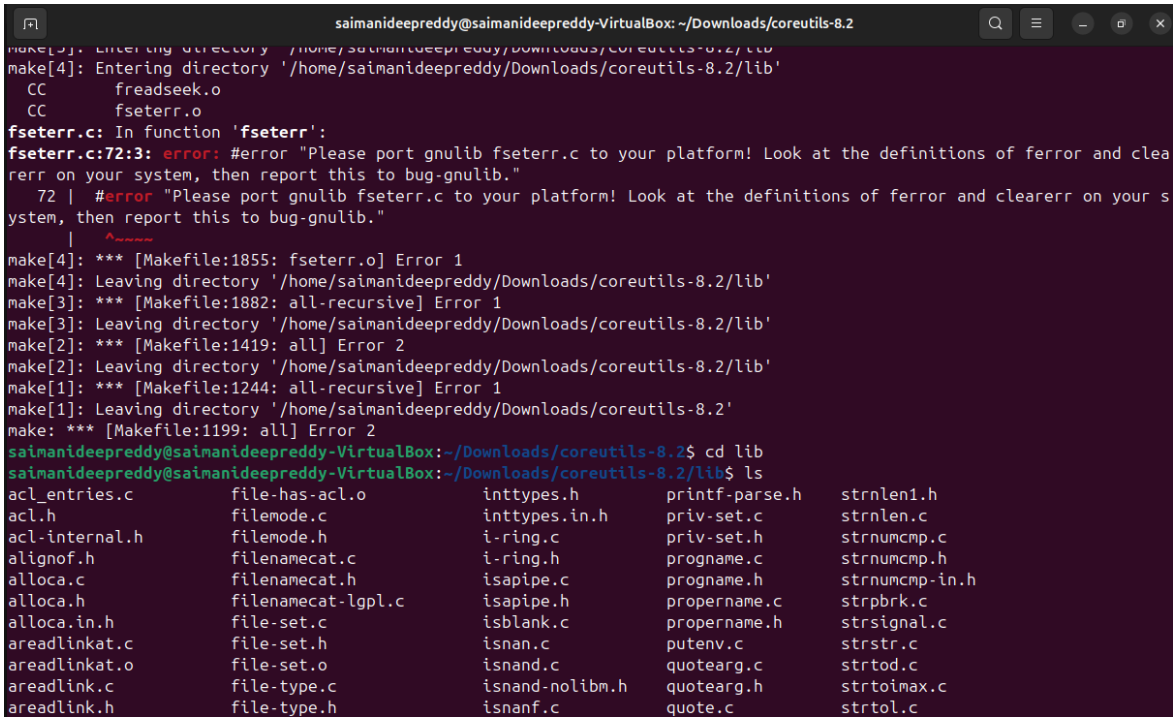

saimanideepreddy@saimanideepreddy-VirtualBox: ~/Downloads/coreutils-8.2
  CC   filenamecat-lgpl.o
  CC   fopen-safer.o
  CC   fprintftime.o
  CC   fpurge.o
  CC   freopen-safer.o
  CC   fseeko.o
fseeko.c: In function 'rpl_fseeko':
fseeko.c:91:4: error: #error "Please port gnulib fseeko.c to your platform! Look at the code in fpurge.c, then report this to bug-gnulib."
   91 | && fp->__put_limit == fp->__bufp
      | ^ ~~
make[4]: *** [Makefile:1855: fseeko.o] Error 1
make[4]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[3]: *** [Makefile:1882: all-recursive] Error 1
make[3]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[2]: *** [Makefile:1419: all] Error 2
make[2]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2/lib'
make[1]: *** [Makefile:1244: all-recursive] Error 1
make[1]: Leaving directory '/home/saimanideepreddy/Downloads/coreutils-8.2'
make: *** [Makefile:1199: all] Error 2
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-8.2$ cd lib
saimanideepreddy@saimanideepreddy-VirtualBox:~/Downloads/coreutils-8.2/lib$ ls
acl_entries.c fdopendir.c idcache.h printf-frexpl.c strnlen1.o
acl.h fdopendir.o ignore-value.h printf-frexpl.h strnlen.c
acl-internal.h fd-reopen.c imaxtostr.c printf-frexpl.o strnumcmp.c
alignof.h fd-reopen.h inet_ntop.c printf-frexp.o strnumcmp.h
alloca.c fd-safer.c intprops.h printf-parse.c strnumcmp-in.h
alloca.h fd-safer.o inttostr.c printf-parse.h strpbrk.c
alloca.in.h fflush.c inttostr.h priv-set.c strsignal.c
areadlinkat.c fflush.o inttypes.h priv-set.h strstr.c
areadlinkat.o fileblocks.c inttypes.in.h priv-set.o strtod.c

In sort.c, the error 'WNOHANG' undeclared (first use in this function) was resolved by including the <sys/wait.h> header file, which defines the WNOHANG macro. This inclusion allowed the waitpid function to properly utilize the WNOHANG option, preventing blocking behavior when there are no child processes to wait for.

In sort.c, the errors related to the implicit declaration of the functions WIFEXITED and WEXITSTATUS were resolved by including the appropriate headers, specifically <sys/wait.h>. Additionally, the functions were properly defined in the code, ensuring that their usage complies with the C standard. This correction enhanced the code's clarity and functionality, preventing potential runtime errors.



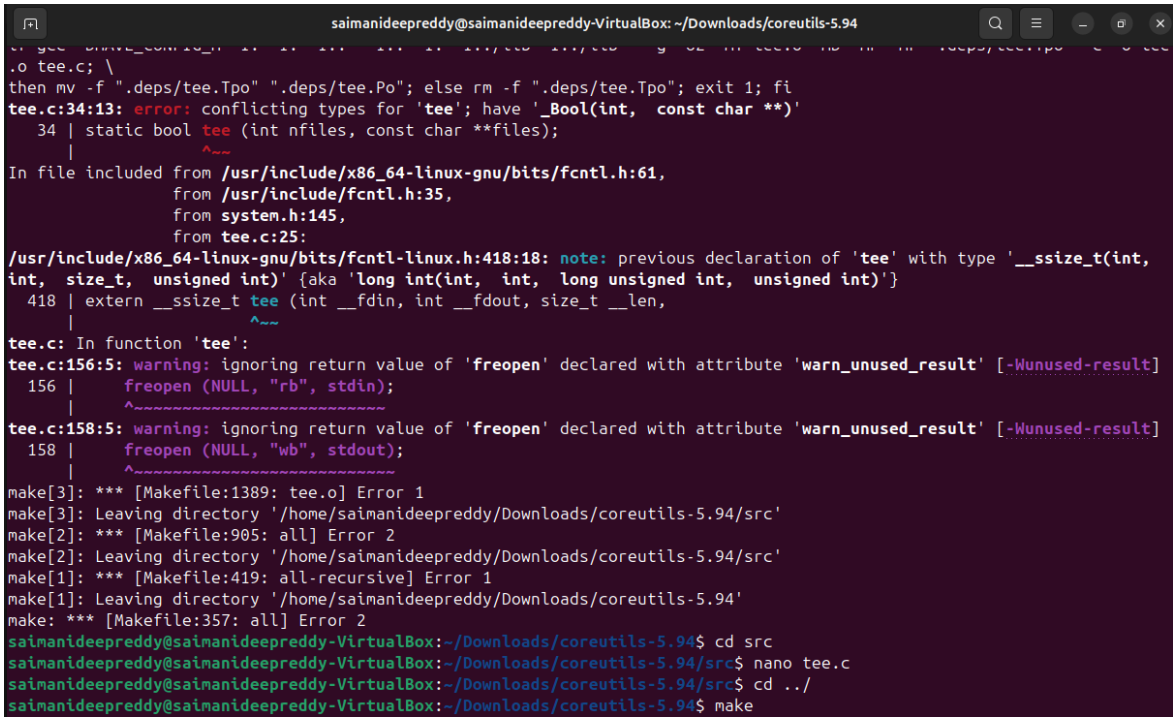
In fseeko.c, the error related to the expression && fp->__put__limit == fp->__bufp suggests issues with accessing members of the FILE structure. To resolve this, it is essential to ensure that the pointer fp is valid before dereferencing it and to confirm that the members __put__limit and __bufp exist in the structure. Additionally, clarifying the expression with parentheses may enhance readability and prevent logical errors.



Challenges in Coreutils version 5.94

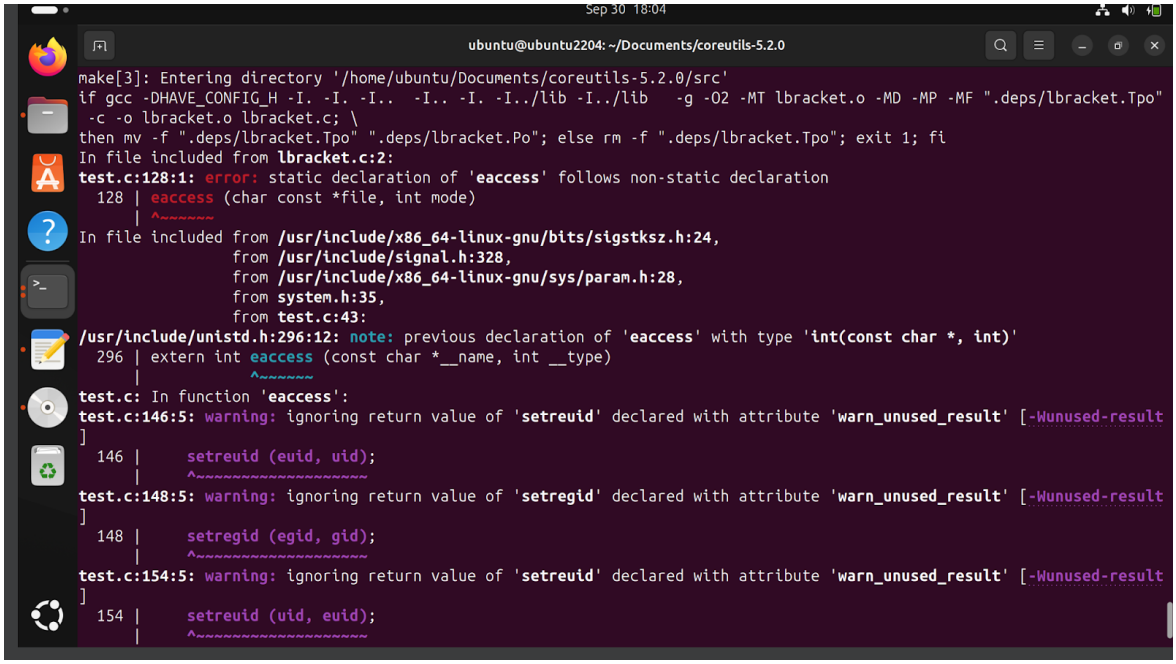
In tee.c, a conflict involving the tee function, originally defined as _Bool(int nfiles, const char **files), was resolved by renaming the function to my_tee. This update prevented interference with the standard library's tee function, ensuring consistent functionality throughout the code.

To address the warning about the return value of freopen, which is flagged by the warn_unused_result attribute, the return value was explicitly checked. Error handling mechanisms were added to manage potential failures, enhancing the code's reliability and aligning it with best practices for robust programming.



Challenges in Coreutils version 5.2.0

In test.c, the error "static declaration of eaccess follows non-static declaration" arose from a conflict between static and non-static definitions of the eaccess function. To resolve this, a custom version of the function was created with the prefix my_, ensuring the declaration was correct. Similar issues were encountered in earlier versions, highlighting the importance of consistent function naming and declaration to avoid such conflicts in the code.



In tee.c, the type conflict involving the tee function, initially defined as (int, const char*), was resolved by renaming it to my_tee. This change eliminated the conflict with the standard tee function, ensuring proper declaration and seamless integration within the code.

2.3. Challenges Encountered

1. Dependency Issues:

Some older coreutils versions required older libraries or specific versions of `glibc`.

- **Error**: "undefined reference to `memcpy'" during compilation.
- **Resolution**: We installed compatible `glibc` versions and used a containerized environment to avoid conflicts with the host system libraries.

2. Compatibility with Modern Systems:

Older versions of coreutils (e.g., 5.2.0) were not fully compatible with the modern Ubuntu 20.04 environment.

- **Error**: "configure: error: cannot compute sizeof (void *)"
- **Resolution**: We resolved this by compiling in a chroot environment or using Docker containers with an older Linux distribution.

3. Conflicts with Installed Utilities:

Installing multiple versions of coreutils on the same system led to conflicts with existing binaries (e.g., `/usr/bin/mv`).

- **Resolution**: We installed each version in a separate directory using the `--prefix` option during configuration:

```
./configure --prefix=/usr/local/coreutils-<version>
     make
     sudo make install
           To use a specific version, the `PATH` was updated
dynamically:
     export PATH=/usr/local/coreutils-<version>/bin:$PATH
```

2.4. Resolving Errors

Each issue faced during the installation process was addressed through careful debugging and configuration changes. For example:

- **Missing Functions:** For functions such as strndup or memrchr that were missing, we updated the source code to add alternative implementations or utilized compatible library versions.
- **Build Failures:** For build failures caused by deprecated tools, we adjusted the Makefile to update outdated flags or dependencies.

2.5. Final Environment Setup

After installation, we verified the functionality of each coreutils version by executing specific binaries (e.g., `mv`, `rm`, `ls`) to ensure they worked as expected. The environment was then used to reproduce known bugs and analyze behavior with the help of the custom PIN Tool script.

GENERATING BUG STRACE FILES AND ANALYZING

Bug Strace File:

- **<u>"mv" Coreutils 6.9:</u>**

```
0x76357b8a59c9: c(0x0, 0x76357b8b7a18, 0x0, 0x37f,
0x0, 0x76357b8b7440) returns:
0xc
Thread 0 - Syscall: brk (number: c)
    Return Value: 12
0x76357b8a6d2a: 9(0x0, 0x2000, 0x3, 0x22,
0xffffffff, 0x0) returns: 0x9
Thread 0 - Syscall: mmap (number: 9)
    Return Value: 9
0x76357b8a68c9: 15(0x76357b8b05b0, 0x4,
0x60e86890eb60, 0x22, 0x76357b881000,
0x76357b8b7440) returns: 0x15
Thread 0 - Syscall: access (number: 15)
    Return Value: 21
0x76357b8a6b6f: 101(0xffffff9c, 0x76357b8af38f,
0x80000, 0x0, 0xffff, 0x0) returns:
0x101
Thread 0 - Syscall: openat (number: 101)
    Return Value: 257
0x76357b8a68f9: 5(0x3, 0x7ffeade990c0, 0x80000, 0x0,
0xffff, 0x0) returns: 0x5
Thread 0 - Syscall: fstat (number: 5)
    Return Value: 5
0x7635682f5b49: 66(0xffffff9c, 0x7ffeade9aa84,
0x7ffeade999a0, 0x0, 0x0,
      0x7ffeade99eb0) returns: 0x66
Thread 0 - Syscall: getuid (number: 66)
    Return Value: 102
0x7635682f4769: 68(0xffffff9c, 0x7ffeade9aa84,
0x7ffeade999a0, 0x0, 0x0,  0x7ffeade99eb0) returns:
0x68
Thread 0 - Syscall: getgid (number: 68)
    Return Value: 104
0x763568316629: 15(0x7ffeade9aa84, 0x2, 0x3e8, 0x0,
0x0, 0x7ffeade99eb0) returns:  0x15
Thread 0 - Syscall: access (number: 15)
    Return Value: 21
0x7635682661c9: 52(0x7ffeade9aa7c, 0x7ffeade9aa84,
0x1, 0x7ffeade99eb0, 0x0, 0x7ffeade99eb0) returns:
0x52
Thread 0 - Syscall: rename (number: 52)
    Return Value: 82
```

```
0x76356831c609: 3(0x1, 0x1, 0x763568401ee0,
0x7ffeade99eb0, 0x0, 0x7ffeade99eb0) returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x76356831c609: 3(0x2, 0x1, 0x763568401ee0,
0x7ffeade99eb0, 0x0, 0x7ffeade99eb0)  returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x7635682ee21b: e7(0x0, 0xffffffffffffff88, 0xe7,
0x7ffeade99d4f, 0x7ffeade99dc0, 0x0)  returns: 0xe7
Thread 0 - Syscall: exit_group (number: e7)
    Return Value: 231
Total Instructions: 245465
```

- **“rm” Coreutils 8.4:**

```
0x7e67e93939c9: c(0x0, 0x7e67e93a5a18, 0x0, 0x37f, 0x0,
0x7e67e93a5440) returns: 0xc
Thread 0 - Syscall: brk (number: c)
    Return Value: 12
0x7e67e9394d2a: 9(0x0, 0x2000, 0x3, 0x22, 0xffffffff,
0x0) returns: 0x9
Thread 0 - Syscall: mmap (number: 9)
    Return Value: 9
0x7e67e93948c9: 15(0x7e67e939e5b0, 0x4, 0x5d612215dc08,
0x22, 0x7e67e936f000, 0x7e67e93a5440) returns: 0x15
Thread 0 - Syscall: access (number: 15)
    Return Value: 21
0x7e67e9394b6f: 101(0xffffff9c, 0x7e67e939d38f, 0x80000,
0x0, 0xffff, 0x0) returns: 0x101
Thread 0 - Syscall: openat (number: 101)
    Return Value: 257
0x7e67e93948f9: 5(0x3, 0x7fffd8cc5f20, 0x80000, 0x0,
0xffff, 0x0) returns: 0x5
Thread 0 - Syscall: fstat (number: 5)
    Return Value: 5
0x7e67d5d173bc: 106(0xffffff9c, 0x5d61230ab800,
0x5d61230ab770, 0x100, 0x7e67d5e03b20, 0x120) returns:
0x106
Thread 0 - Syscall: newfstatat (number: 106)
    Return Value: 262
0x7e67d5d1c439: 107(0xffffff9c, 0x5d61230aa4e0, 0x0, 0x0,
0x3, 0x0) returns: 0x107
Thread 0 - Syscall: unlinkat (number: 107)
    Return Value: 263
0x7e67d5d1afd9: 8(0x0, 0x0, 0x1, 0x5d61230aa450, 0x0,
0x7) returns: 0x8
Thread 0 - Syscall: lseek (number: 8)
    Return Value: 8
```

```
0x7e67d5d1c609: 3(0x0, 0x1, 0x7e67d5e01ee0,
0x5d61230aa450, 0x0, 0x7) returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x7e67d5d1c609: 3(0x1, 0x1, 0x7e67d5e01ee0,
0x5d61230aa450, 0x0, 0x7) returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x7e67d5d1c609: 3(0x2, 0x1, 0x7e67d5e01ee0,
0x5d61230aa450, 0x0, 0x7) returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x7e67d5cee21b: e7(0x0, 0xffffffffffffff88, 0xe7,
0x7fffd8cc6bdf, 0x7fffd8cc6c50, 0x0) returns: 0xe7
Thread 0 - Syscall: exit_group (number: e7)
    Return Value: 231
Total Instructions: 237031
```

“mknod” Coreutils 5.2.1:

```
0x70504c74d9c9: c(0x0, 0x70504c75fa18, 0x0, 0x37f, 0x0,
0x70504c75f440) returns: 0xc
Thread 0 - Syscall: brk (number: c)
    Return Value: 12
0x70504c74ed2a: 9(0x0, 0x2000, 0x3, 0x22, 0xffffffff,
0x0) returns: 0x9
Thread 0 - Syscall: mmap (number: 9)
    Return Value: 9
0x70504c74e8c9: 15(0x70504c7585b0, 0x4, 0x5abb2c9dad00,
0x22, 0x70504c729000, 0x70504c75f440) returns: 0x15
Thread 0 - Syscall: access (number: 15)
    Return Value: 21
0x70504c74eb6f: 101(0xffffff9c, 0x70504c75738f, 0x80000,
0x0, 0xffff, 0x0) returns: 0x101
Thread 0 - Syscall: openat (number: 101)
    Return Value: 257
0x70504c74e8f9: 5(0x3, 0x7ffcb2691940, 0x80000, 0x0,
0xffff, 0x0) returns: 0x5
Thread 0 - Syscall: fstat (number: 5)
    Return Value: 5
0x70503911d779: c(0x5abb2d9e6000, 0x705039203b20,
0x70503920ad70, 0x1, 0x21000, 0x0) returns: 0xc
Thread 0 - Syscall: brk (number: c)
    Return Value: 12
0x70503911c8b2: 101(0xffffff9c, 0x7050391d48b0, 0x80000,
0x0, 0x5abb2d9c5010, 0x7) returns: 0x101
Thread 0 - Syscall: openat (number: 101)
    Return Value: 257
```

```
0x705039117389: 5(0x3, 0x705039204800, 0x0, 0x0,
0x5abb2d9c5010, 0x7) returns: 0x5
Thread 0 - Syscall: fstat (number: 5)
    Return Value: 5
0x70503912531a: 9(0x0, 0x574500, 0x1, 0x2, 0x3, 0x0)
returns: 0x9
Thread 0 - Syscall: mmap (number: 9)
    Return Value: 9
0x70503911c609: 3(0x3, 0x574500, 0x2ba0, 0x2, 0x3, 0x0)
returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x70503911b0e5: 103(0xffffff9c, 0x7ffcb2693a7e, 0x11b6,
0x0, 0x0, 0x0) returns: 0x103
Thread 0 - Syscall: mknodat (number: 103)
    Return Value: 259
0x7050390ee21b: e7(0x0, 0xffffffffffffff88, 0xe7,
0x7ffcb269269f, 0x7ffcb2692710, 0x0) returns: 0xe7
Thread 0 - Syscall: exit_group (number: e7)
    Return Value: 231
Total Instructions: 211243
```

“mkfifo” Coreutils 5.2.1:

```
0x78bd27d4f9c9: c(0x0, 0x78bd27d61a18, 0x0, 0x37f, 0x0,
0x78bd27d61440) returns: 0xc
Thread 0 - Syscall: brk (number: c)
    Return Value: 12
0x78bd27d50d2a: 9(0x0, 0x2000, 0x3, 0x22, 0xffffffff,
0x0) returns: 0x9
Thread 0 - Syscall: mmap (number: 9)
    Return Value: 9
0x78bd27d508c9: 15(0x78bd27d5a5b0, 0x4, 0x6235e34a8d00,
0x22, 0x78bd27d2b000, 0x78bd27d61440) returns: 0x15
Thread 0 - Syscall: access (number: 15)
    Return Value: 21
0x78bd27d50b6f: 101(0xffffff9c, 0x78bd27d5938f, 0x80000,
0x0, 0xffff, 0x0) returns: 0x101
Thread 0 - Syscall: openat (number: 101)
    Return Value: 257
0x78bd27d508f9: 5(0x3, 0x7fff0d76edb0, 0x80000, 0x0,
0xffff, 0x0) returns: 0x5
Thread 0 - Syscall: fstat (number: 5)
    Return Value: 5
0x78bd1471d779: c(0x6235e4404000, 0x78bd14803b20,
0x78bd1480ad70, 0x1, 0x21000, 0x0) returns: 0xc
Thread 0 - Syscall: brk (number: c)
    Return Value: 12
0x78bd1471c8b2: 101(0xffffff9c, 0x78bd147d48b0, 0x80000,
0x0, 0x6235e43e3010, 0x7) returns: 0x101
```

```
Thread 0 - Syscall: openat (number: 101)
    Return Value: 257
0x78bd14717389: 5(0x3, 0x78bd14804800, 0x0, 0x0,
0x6235e43e3010, 0x7) returns: 0x5
Thread 0 - Syscall: fstat (number: 5)
    Return Value: 5
0x78bd1472531a: 9(0x0, 0x574500, 0x1, 0x2, 0x3, 0x0)
returns: 0x9
Thread 0 - Syscall: mmap (number: 9)
    Return Value: 9
0x78bd1471c609: 3(0x3, 0x574500, 0x2ba0, 0x2, 0x3, 0x0)
returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x78bd1471b0e5: 103(0xffffff9c, 0x7fff0d770a7f, 0x11b6,
0x0, 0x0, 0x0) returns: 0x103
Thread 0 - Syscall: mknodat (number: 103)
    Return Value: 259
0x78bd146ee21b: e7(0x0, 0xffffffffffffff88, 0xe7,
0x7fff0d76fb2f, 0x7fff0d76fba0, 0x0) returns: 0xe7
Thread 0 - Syscall: exit_group (number: e7)
    Return Value: 231
Total Instructions: 209784
```

- **<u>"mkdir" Coreutils 5.2.1:</u>**

```
0x756bcd8d09c9: c(0x0, 0x756bcd8e2a18, 0x0, 0x37f, 0x0,
0x756bcd8e2440) returns: 0xc
Thread 0 - Syscall: brk (number: c)
    Return Value: 12
0x756bcd8d1d2a: 9(0x0, 0x2000, 0x3, 0x22, 0xffffffff,
0x0) returns: 0x9
Thread 0 - Syscall: mmap (number: 9)
    Return Value: 9
0x756bcd8d18c9: 15(0x756bcd8db5b0, 0x4, 0x5af5f8400ce0,
0x22, 0x756bcd8ac000, 0x756bcd8e2440) returns: 0x15
Thread 0 - Syscall: access (number: 15)
    Return Value: 21
0x756bcd8d1b6f: 101(0xffffff9c, 0x756bcd8da38f, 0x80000,
0x0, 0xffff, 0x0) returns: 0x101
Thread 0 - Syscall: openat (number: 101)
    Return Value: 257
0x756bcd8d18f9: 5(0x3, 0x7fff19792350, 0x80000, 0x0,
0xffff, 0x0) returns: 0x5
Thread 0 - Syscall: fstat (number: 5)
    Return Value: 5
0x756bba317389: 5(0x3, 0x756bba404800, 0x0, 0x0,
0x5af5fa143010, 0x7) returns: 0x5
Thread 0 - Syscall: fstat (number: 5)
```

```
    Return Value: 5
0x756bba32531a: 9(0x0, 0x574500, 0x1, 0x2, 0x3, 0x0)
returns: 0x9
Thread 0 - Syscall: mmap (number: 9)
    Return Value: 9
0x756bba31c609: 3(0x3, 0x574500, 0x2ba0, 0x2, 0x3, 0x0)
returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x756bba31c3f9: 5f(0x0, 0x7fff19794f16, 0x756bba403240,
0x5af5f83fe5de, 0x0, 0x0) returns: 0x5f
Thread 0 - Syscall: umask (number: 5f)
    Return Value: 95
0x756bba31c3f9: 5f(0x2, 0x7fff19794f16, 0x756bba403240,
0x5af5f83fe5de, 0x0, 0x0) returns: 0x5f
Thread 0 - Syscall: umask (number: 5f)
    Return Value: 95
0x756bba31b029: 53(0x7fff19793a7c, 0x1ff, 0x1ff,
0x5af5f83fe5de, 0x0, 0x0) returns: 0x53
Thread 0 - Syscall: mkdir (number: 53)
    Return Value: 83
0x756bba2ee21b: e7(0x0, 0xffffffffffffff88, 0xe7,
0x7fff197930af, 0x7fff19793120, 0x0) returns: 0xe7
Thread 0 - Syscall: exit_group (number: e7)
    Return Value: 231
Total Instructions: 212813
```

- **<u>“ln” Coreutils 5.94:</u>**

```
0x7dbad9b969c9: c(0x0, 0x7dbad9ba8a18, 0x0, 0x37f, 0x0,
0x7dbad9ba8440) returns: 0xc
Thread 0 - Syscall: brk (number: c)
    Return Value: 12
0x7dbad9b97d2a: 9(0x0, 0x2000, 0x3, 0x22, 0xffffffff,
0x0) returns: 0x9
Thread 0 - Syscall: mmap (number: 9)
    Return Value: 9
0x7dbad9b978c9: 15(0x7dbad9ba15b0, 0x4, 0x5c3b61f3ecc0,
0x22, 0x7dbad9b72000, 0x7dbad9ba8440) returns: 0x15
Thread 0 - Syscall: access (number: 15)
    Return Value: 21
0x7dbad9b97b6f: 101(0xffffff9c, 0x7dbad9ba038f, 0x80000,
0x0, 0x0, 0x0) returns: 0x101
Thread 0 - Syscall: openat (number: 101)
    Return Value: 257
0x7dbad9b978f9: 5(0x3, 0x7ffd29099bb0, 0x80000, 0x0, 0x0,
0x0) returns: 0x5
Thread 0 - Syscall: fstat (number: 5)
    Return Value: 5
```

```
0x7dbac652531a: 9(0x0, 0x574500, 0x1, 0x2, 0x3, 0x0)
returns: 0x9
Thread 0 - Syscall: mmap (number: 9)
    Return Value: 9
0x7dbac651c609: 3(0x3, 0x574500, 0x2ba0, 0x2, 0x3, 0x0)
returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x7dbac65173bc: 106(0xffffff9c, 0x7ffd2909ba73,
0x7ffd2909a9e0, 0x0, 0x1000, 0x0) returns: 0x106
Thread 0 - Syscall: newfstatat (number: 106)
    Return Value: 262
0x7dbac65173bc: 106(0xffffff9c, 0x7ffd2909ba62,
0x7ffd2909a840, 0x100, 0x1000, 0x0) returns: 0x106
Thread 0 - Syscall: newfstatat (number: 106)
    Return Value: 262
0x7dbac651ae59: 56(0x7ffd2909ba62, 0x7ffd2909ba73,
0x7ffd2909a840, 0x100, 0x1000, 0x0) returns: 0x56
Thread 0 - Syscall: link (number: 56)
    Return Value: 86
0x7dbac651c609: 3(0x1, 0x1, 0x7dbac6601ee0, 0x100, 0x0,
0x0) returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x7dbac64ee21b: e7(0x0, 0xffffffffffffff88, 0xe7,
0x7ffd2909a86f, 0x7ffd2909a8e0, 0x0) returns: 0xe7
Thread 0 - Syscall: exit_group (number: e7)
    Return Value: 231
Total Instructions: 217600
```

- **“chmod” Coreutils 8.2:**

```
0x761b00ead9c9: c(0x0, 0x761b00ebfa18, 0x0, 0x37f, 0x0,
0x761b00ebf440) returns: 0xc
Thread 0 - Syscall: brk (number: c)
    Return Value: 12
0x761b00eaed2a: 9(0x0, 0x2000, 0x3, 0x22, 0xffffffff,
0x0) returns: 0x9
Thread 0 - Syscall: mmap (number: 9)
    Return Value: 9
0x761b00eae8c9: 15(0x761b00eb85b0, 0x4, 0x5e2bd8c1bc88,
0x22, 0x761b00e89000, 0x761b00ebf440) returns: 0x15
Thread 0 - Syscall: access (number: 15)
    Return Value: 21
0x761b00eaeb6f: 101(0xffffff9c, 0x761b00eb738f, 0x80000,
0x0, 0xffff, 0x0) returns: 0x101
Thread 0 - Syscall: openat (number: 101)
    Return Value: 257
```

```
0x761b00eae8f9: 5(0x3, 0x7ffc93f293a0, 0x80000, 0x0,
0xffff, 0x0) returns: 0x5
Thread 0 - Syscall: fstat (number: 5)
    Return Value: 5
0x761aed91c609: 3(0x3, 0x574500, 0x2ba0, 0x2, 0x3, 0x0)
returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x761aed91c3f9: 5f(0x0, 0x20, 0x0, 0x5e2bd95bc450,
0x761aeda03b20, 0x30) returns: 0x5f
Thread 0 - Syscall: umask (number: 5f)
    Return Value: 95
0x761aed9173bc: 106(0xffffff9c, 0x5e2bd95bd830,
0x5e2bd95bd7a0, 0x0, 0x761aeda03b20, 0x120) returns:
0x106
Thread 0 - Syscall: newfstatat (number: 106)
    Return Value: 262
0x761aed916e45: 10c(0xffffff9c, 0x5e2bd95bc510, 0x1b4,
0x1b4, 0x0, 0x4000) returns: 0x10c
Thread 0 - Syscall: fchmodat (number: 10c)
    Return Value: 268
0x761aed91c609: 3(0x1, 0x1, 0x761aeda01ee0,
0x5e2bd95bc480, 0x0, 0x7) returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x761aed91c609: 3(0x2, 0x1, 0x761aeda01ee0,
0x5e2bd95bc480, 0x0, 0x7) returns: 0x3
Thread 0 - Syscall: close (number: 3)
    Return Value: 3
0x761aed8ee21b: e7(0x0, 0xffffffffffffff88, 0xe7,
0x7ffc93f2a03f, 0x7ffc93f2a0b0, 0x0) returns: 0xe7
Thread 0 - Syscall: exit_group (number: e7)
    Return Value: 231
Total Instructions: 226429
```

CONCLUSION

This project demonstrated how Intel's PIN Tool can be utilized to trace system calls made by various versions of the Linux core utilities (coreutils). By creating strace.out files, we were able to gather comprehensive insights into how these utilities interact with the operating system kernel. This approach proved effective for examining past bugs and understanding system call behaviors in different setups.

The installation and management of multiple coreutils versions revealed common challenges in software versioning and compatibility, including dependency issues and environmental discrepancies. These were addressed through precise debugging, resolving dependencies, and using isolated environments, illustrating the value of a methodical approach to analyzing system software.

In doing so, we developed a repeatable methodology for tracing system calls and analyzing software behavior in detail. This methodology has broad applications in debugging, performance enhancement, and security evaluation. Ultimately, the project highlights how tools like PIN Tool can significantly enhance the visibility and understanding of complex system operations, leading to improved software reliability and long-term maintainability.

FUTURE WORK

This project lays the foundation for further exploration into system call analysis and debugging using dynamic instrumentation tools. Several directions for future work can be pursued to expand upon the outcomes of this project:

- **Automated Bug Detection and Classification**

Develop automated scripts or tools to detect, classify, and analyze patterns in system calls that may lead to specific bugs or performance bottlenecks. Machine learning techniques could be explored for this purpose.

- **Dynamic Analysis Across More Versions and Utilities**

Extend the scope of analysis to include additional versions of coreutils and other system utilities to build a comprehensive dataset of system call behavior over time. This could help identify long-term trends or regressions.

- **Dynamic Analysis Across More Versions and Utilities**

Extend the scope of analysis to include additional versions of coreutils and other system utilities to build a comprehensive dataset of system call behavior over time. This could help identify long-term trends or regressions.

- **Integration with Modern Tracing Tools**

Combine the capabilities of PIN Tool with modern observability tools like eBPF (extended Berkeley Packet Filter) to enhance performance tracing and reduce overhead during system call instrumentation.

- **Real-time Tracing and Analysis**

Adapt the tracing methodology for real-time analysis of system calls in production environments. This could have significant implications for debugging and monitoring live systems with minimal disruption.

- **Comparative Analysis Across Platforms**

Investigate the behavior of core utilities on different Linux distributions, kernel versions, or even other operating systems like BSD or macOS, to understand platform-specific variations.

- **Enhanced Visualization**

Develop visualization tools to represent system call traces in a more intuitive manner, such as graphs showing dependencies, call frequencies, or execution timelines. This would aid in faster analysis and reporting.

- **Security Applications**

  Extend the methodology to study potential security vulnerabilities, such as system call misuse or privilege escalations, within utilities. This could be critical for hardening system utilities against exploits.

- **Performance Optimization Studies**

  Use the insights from the tracing data to identify inefficiencies in system utilities and propose optimizations, contributing to improved software performance and reduced resource usage.

By pursuing these directions, the methodology established in this project can evolve into a broader framework for analyzing, debugging, and optimizing software across diverse computing environments.

REFERENCES

[1] Version 5.0.4.rh65. Red Hat Bugzilla – Main Page. https://bugzilla.redhat.com/

[2] Gautam Altekar and Ion Stoica. 2009. ODR: Output-deterministic Replay for Multicore Debugging. In Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (Big Sky, Montana, USA) (SOSP '09). Association for Computing Machinery, New York, NY, USA, 193–206. https://doi.org/10.1145/ 1629575.1629594

[3] Darren O. Benham. 1997. Debian Bug Tracking System. https://www.debian.org/ Bugs/

[4] Darren O. Benham. 1997. GNU Bug Tracker. https://debbugs.gnu.org/

[5] Jong-Deok Choi and Andreas Zeller. 2002. Isolating Failure-Inducing Thread Schedules (ISSTA '02). Association for Computing Machinery, New York, NY, USA, 210–220. https://doi.org/10.1145/566172.566211

[6] Holger Cleve and Andreas Zeller. 2005. Locating Causes of Program Failures. In Proceedings of the 27th International Conference on Software Engineering (St. Louis, MO, USA) (ICSE '05). Association for Computing Machinery, New York, NY, USA, 342–351. https://doi.org/10.1145/1062455.1062522

[7] Geeks for Geeks. 2019. Stress Testing | Software Testing. https://www. geeksforgeeks.org/stress-testing-software-testing/

[8] Thomas Hamilton. 2021. What is STRESS Testing in Software Testing? Tools, Types, Examples. https://www.guru99.com/stress-testing-tutorial.html

[9] Intel. 2004. Pin 3.21 User Guide. https://software.intel.com/sites/landingpage/ pintool/docs/98484/Pin/html/

[10] Oren Laadan, Chia-Che Tsai, Nicolas Viennot, Chris Blinn, Peter Senyao Du, and Jason Nieh. 2011. Finding Concurrency Errors in Sequential Code—OS-level, In-vivo Model Checking of Process Races. In 13th Workshop on Hot Topics in Operating Systems (HotOS XIII). USENIX Association, Napa, CA. https://www.usenix.org/conference/hotosxiii/finding-concurrency-errorssequential-code{\T1\textemdash}os-level-vivo-model-checking-process

[11] Oren Laadan, Nicolas Viennot, Chia-Che Tsai, Chris Blinn, Junfeng Yang, and Jason Nieh. 2011. Pervasive Detection of Process Races in Deployed Systems. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (Cascais, Portugal) (SOSP '11). Association for Computing Machinery, New York, NY, USA, 353–367. https://doi.org/10.1145/2043556.2043589

[12] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazelwood. 2005. Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation. SIGPLAN Not. 40, 6 (June 2005), 190–200. https://doi.org/10.1145/1064978.1065034

[13] Madanlal Musuvathi and Shaz Qadeer. 2007. Iterative Context Bounding for Systematic Testing of Multithreaded Programs. In Proceedings of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation (San Diego, California, USA) (PLDI '07). Association for Computing Machinery, New York, NY, USA, 446–455. https://doi.org/10.1145/1250734.1250785

[14] Nicholas Nethercote. 2004. Dynamic Binary Analysis and Instrumentation. Technical Report UCAM-CL-TR-606. University of Cambridge, Computer Laboratory. https://doi.org/10.48456/tr-606

[15] Robert H. B. Netzer and Barton P. Miller. 1992. What Are Race Conditions? Some Issues and Formalizations. ACM Lett. Program. Lang. Syst. 1, 1 (March 1992), 74–88. https://doi.org/10.1145/130616.130623

[16] Sangmin Park, Richard W. Vuduc, and Mary Jean Harrold. 2010. Falcon: Fault Localization in Concurrent Programs. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1 (Cape Town, South Africa) (ICSE '10). Association for Computing Machinery, New York, NY, USA, 245–254. https://doi.org/10.1145/1806799.1806838

[17] James Ralston. 2008. Bug 438076 - mv Creates Race Condition by Calling unlink() Before rename(). https://bugzilla.redhat.com/show_bug.cgi?id=438076

[18] Takamitsu Tahara, Katsuhiko Gondow, and Seiya Ohsuga. 2008. DRACULA: Detector of Data Races in Signals Handlers. In 2008 15th Asia-Pacific Software Engineering Conference (Beijing, China). 17–24. https://doi.org/10.1109/APSEC. 2008.25

[19] Guy Tchamgoue, Ok-Kyoon Ha, Kyong Hoon Kim, and Yong-Kee Jun. 2011. Lightweight Labeling Scheme for On-the-fly Race Detection of Signal Handlers. Ubiquitous Computing Multimedia Appl. 151, 201–208. https://doi.org/10.1007/ 978-3-642-20998-7_25

[20] Dan Tsafrir, Tomer Hertz, David Wagner, and Dilma Da Silva. 2008. Portably Solving File TOCTTOU Races with Hardness Amplification. In Proceedings of the 6th USENIX Conference on File and Storage Technologies (San Jose, California) (FAST'08). USENIX Association, USA, Article 13, 18 pages.

[21] Tingting Yu, Witawas Srisa-an, and Gregg Rothermel. 2013. SimRacer: An Automated Framework to Support Testing for Process-level Races. In Proceedings of the 2013 International Symposium on Software Testing and Analysis (Lugano, Switzerland) (ISSTA 2013). Association for Computing Machinery, New York, NY, USA, 167–177. https://doi.org/10.1145/2483760.2483771

[22] Tingting Yu, Tarannum S. Zaman, and Chao Wang. 2017. DESCRY: Reproducing System-level Concurrency Failures (ESEC/FSE 2017 (Paderborn, Germany)). Association for Computing Machinery, New York, NY, USA, 694–704. https://doi.org/10.1145/3106237.3106266

[23] Tarannum Shaila Zaman, Xue Han, and Tingting Yu. 2019. SCMiner: Localizing System-level Concurrency Faults from Large System Call Traces (ASE '19). IEEE Press, San Diego, CA, USA, 515–526. https://doi.org/10.1109/ASE.2019.00055

[24] Tarannum Shaila Zaman and Tingting Yu. 2018. Extracting Implicit Programming Rules: Comparing Static and Dynamic Approaches. In Proceedings of the 7th International Workshop on Software Mining (Montpellier, France) (SoftwareMining 2018). Association for Computing Machinery, New York, NY, USA, 1–7. https: //doi.org/10.1145/3242887.3242889

[25] Wei Zhang, Junghee Lim, Ramya Olichandran, Joel Scherpelz, Guoliang Jin, Shan Lu, and Thomas Reps. 2011. ConSeq: Detecting Concurrency Bugs through Sequential Errors (ASPLOS XVI). Association for Computing Machinery, New York, NY, USA, 251–264. https://doi.org/10.1145/1950365.1950395