Debugging and Version control

15-213 / 18-213: Introduction to Computer Systems
12th Recitation, Nov. 14, 2011

Slides by: Lin Xiao(lxiao)
Today

- Debugging with GDB and core file
- Attach GDB to running process
- Heap consistency checking in glibc
- Version control with Git
Debug with core dump file

- Compile your program with option \-g
  - \-g provides debugging information that gdb can use
- In csh:
  - unlimit coredumpsize

- Core dump: contains state of the process when it crashes
- E.g. if a program compiled with \-g option gets segfault, it generates a core dump file
Example code : faulty.c

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv)
{
    char * buf;
    buf = NULL; //obvious and silly mistake
    fgets(buf, 1024, stdin);
    printf("%s\n", buf);
    return 0;
}
Compile and run the program

$ gcc –Wall –g –o faulty faulty.c
$./faulty
1
Segmentation fault (core dumped)

- A core dump file called core.31747/core is generated
- Use gdb to debug the program with the core file
- Then you can examine the state when process crashes
Use gdb with a core file

$ gdb faulty core.31747
GNU gdb Fedora (6.8-29.fc10)
........
Core was generated by `./faulty'.
Program terminated with signal 11, Segmentation fault.
[New process 31747]
#0 0x000000327f869a0e in _IO_getline_info_internal () from /lib64/libc.so.6
  Missing separate debuginfos, use: debuginfo-install glibc-2.9-2.x86_64
  (gdb) bt
#0 0x000000327f869a0e in _IO_getline_info_internal () from /lib64/libc.so.6
#1 0x000000327f8687a7 in fgets () from /lib64/libc.so.6
#2 0x00000000000400578 in main (argc=1, argv=0x7fffaf3c1998) at fault.c:8
  (gdb)
Today

- Debugging with GDB core file
- Attach GDB to running process
- Heap consistency checking in glibc
- Version control with Git
Attaching to a running process

- Process gets stuck (infinite loop)
- Look at status for long running program

- `gdb program process-id`
- `in gdb`
  - `(gdb) attach process-id`

- **How to find process-id**
  - If the process starts in background, the process id is printed
  - Use “`ps aux | grep program`”
  - `man ps`
Today

- Debugging with GDB core file
- Attach GDB to running process
- Heap consistency checking in glibc
- Version control with Git
Heap consistency checking in glibc

- Ask malloc to check the heap consistency by using mcheck
- GNU extension, declared in malloc.h
- `int mcheck (void (*abortfn) (enum mcheck_status status))`
  - Call abortfn when inconsistency is found

- Or set the environment variable `MALLOCCHECK`
- Check and guard against bugs when using `malloc`, `realloc`, `free`
- If `MALLOCCHECK` is set, a special (less efficient) implementation is used to tolerate simple errors
Today

- Debugging with GDB with core file
- Attach GDB to running process
- Heap consistency checking in glibc
- Version control with Git
Version control

- Track and control changes to a project’s files
  - Keep multiple versions
  - Labels/Comments help to identify changes
- Commonly used for team collaboration

- Version control systems:
  - CVS, SVN, etc...
  - We’ll demonstrate how to use Git today
Git overview

- Developed by Linux kernel creator Linus Torvalds
- A distributed versioning file system
  - We only use it with local repository in the recitation
- Installed in shark machines
- “git” lists most commonly used git commands
Create your repository

- Creating a new repository
  - `git init`: Create an empty git repository in current directory
  - `git init malloclab-handout`: specify the directory

- Directory `.git` is created and stores the whole repository content
- **working tree**: project files in the repository
- **index**: snapshot for your project files
Add changes

- Add changes to stage area before commit
- `git add .`
  - Add files in the current directory
- `git add mm.c`
  - Even if mm.c is under version control
- Different from other version control systems: once the file is in version control, you don’t need to add it again)
Commit

- Commit your changes
- `git commit -m "my first commit"
- Each commit is assigned a SHA-1 hash

- If only `mm.c` is changed, you can commit the change by:
  - `git add mm.c`
  - `git commit -m "Implement implicit lists"
- `git commit mm.c -m "Implement implicit lists"
- `git commit -a -m "Implement implicit lists"
Withdraw changes:

- If you haven’t added mm.c to index yet:
  - `git checkout mm.c`

- If mm.c is added to index but not committed yet:
  - `git reset HEAD mm.c`
  - `git checkout mm.c`
Other commands

- **git status**: Show the working tree status
  - # Changes to be committed:
  - # Changed but not updated:
  - # Untracked files:

- **git log**: Show commit logs

- **git tag**
- **git branch**
- **git revert**
Git references

- Git cheat sheets
- Git Tutorial
- git magic