CS 3650 – Computer Systems
Spring 2024
Peter Desnoyers

Lecture 10, Thur Feb 8 2024

Gradescope errors

"cannot run" ← compile failed
anything that looks like it has part of an ASAN error
your debug printfs

@139 ← specification

debugger - gdb

gdb exe
run [arguments]
    ⋮
  <crash>

(gdb) p ← print
       l ← list
       up, down
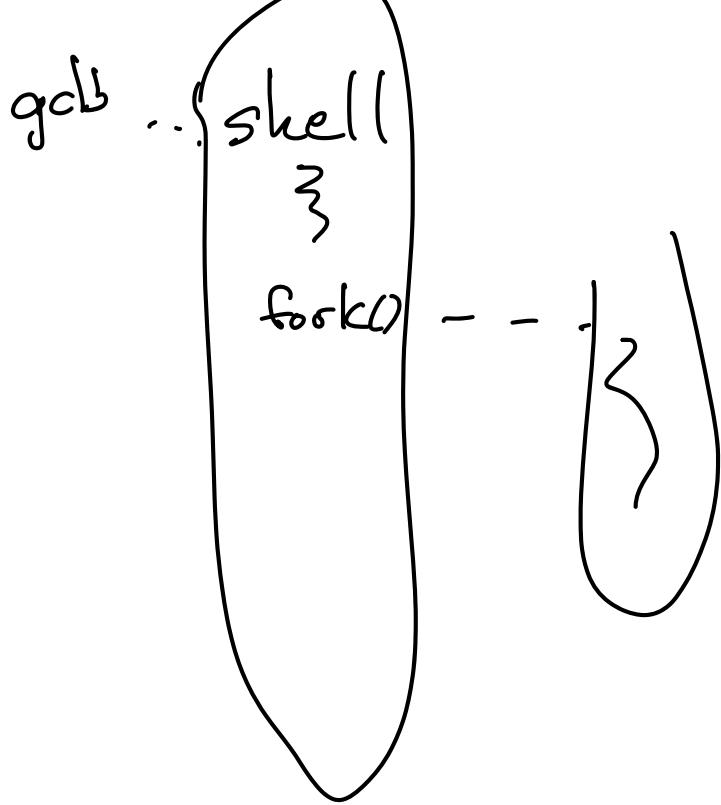   list # ← line #

<return> ← repeat last
b function / line #
       ← break
info break ← list them
delete # ← delete brkt #
n next
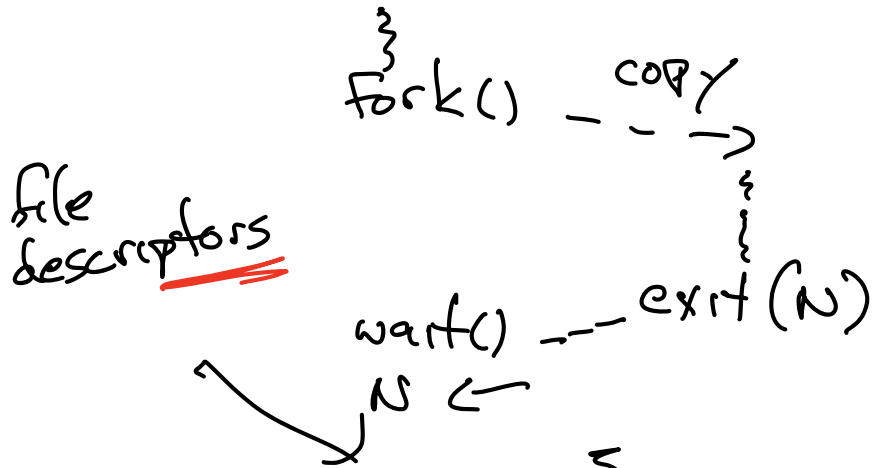s step

gdb ...{shell
      }
      fork() - - - {
                     }

if (pcd ==0) {
~~
    ←
~~
}

default: follow
        parent

set follow-fork-mode child
→ then set break in child

Recap of process/file syscalls

P: use PATH

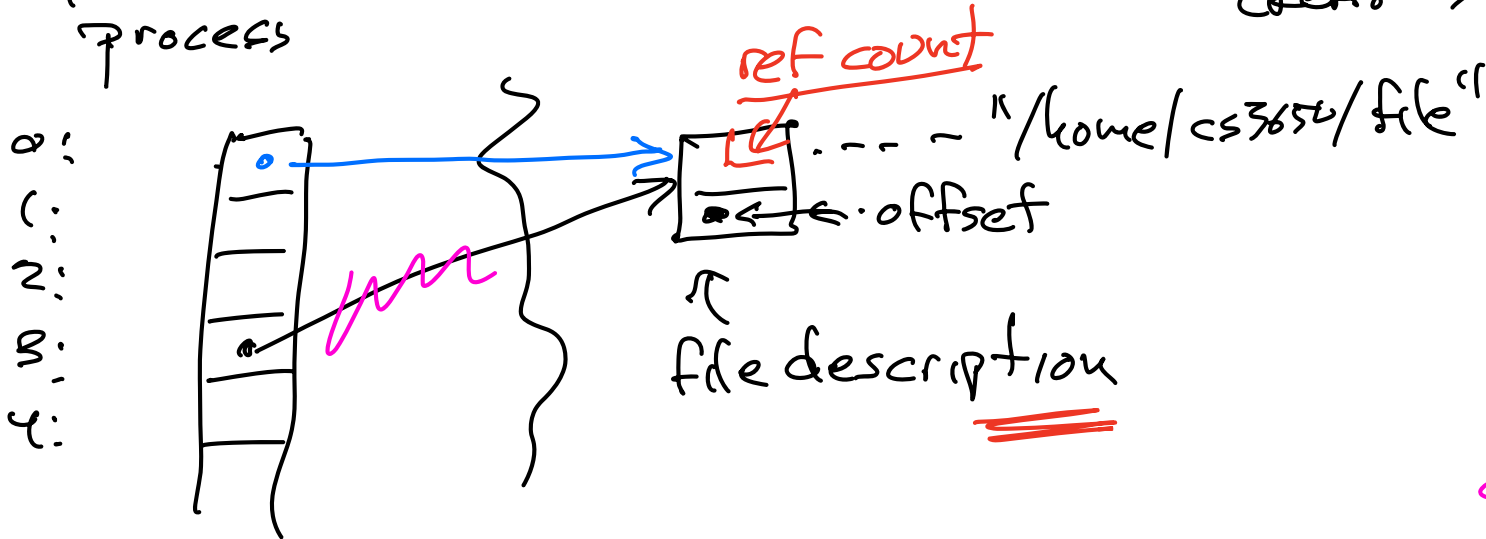execvp (cmd, argu)

char * cmd   e.g. "ls"

char * argv[]

e.s.

fork() --- COPY -->

file descriptors

wait() --- exit(N)

N <---
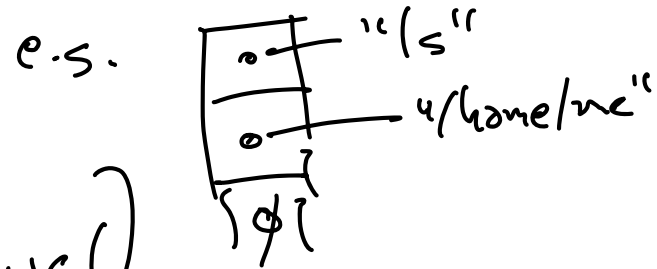
int fd = open("file", O_RDONLY)

fd2 =   (file2", O_WRONLY | TRUNC |
                              CREAT)

Ter-
process

ref count

"/home/cs3650/file"

. offset

file description

0:
1:
2:
3:
4:

dup2(3, 0)
       fd

close(3)

parent

0

3

term

dup2(fd?)

child

0

3

fd=open

close(fd)

redirs

$$\{$$
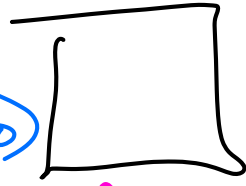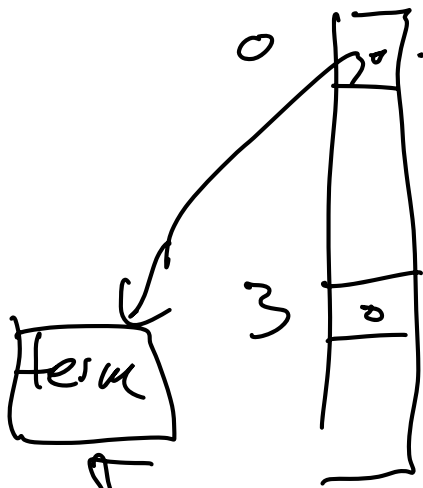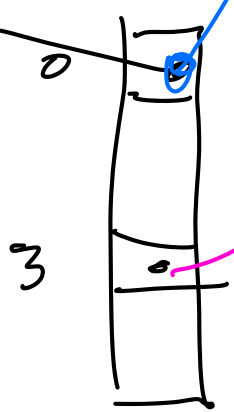
if fork == 0 $\{$

fd = open
dup2(fd, 0)
close(fd)

~ same for 1

$\}$

dup2(fd, 1)

scheduling processes

process 1

scheduler
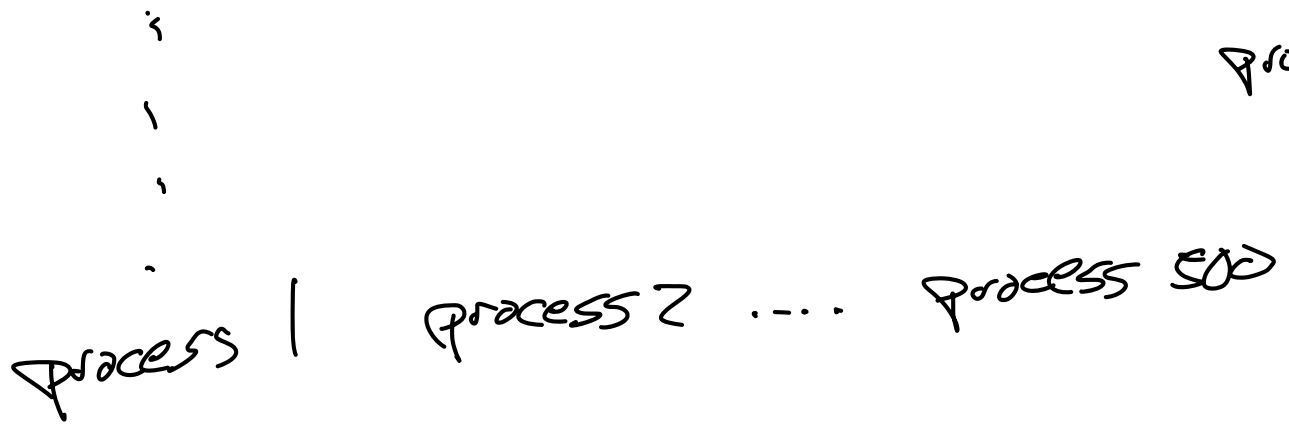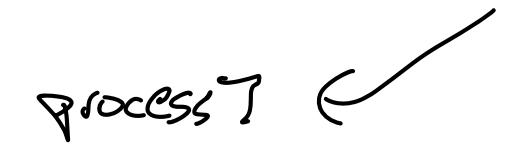
process 7

...

process 1   process 2 .... process 500

function calls   X86_64

ABl ⟵ application Binary Intface

A ⟶ CALL <addr>   PUSH A   MOV me ⟶ RAX
       ⟸   GOTO <addr>   RET   POP ⟶ tmp
return value: GOTO tmp
   RAX

let x = F( ) {
          .. return;
        }

base   steck
Frame   ptr
  Bptr

registess
RAX RBX RCX RDX RDI RSI RBP RSP R8.. R15
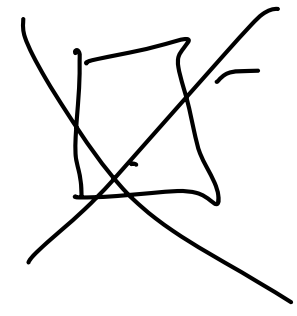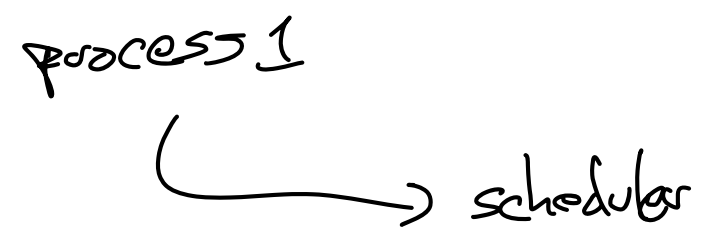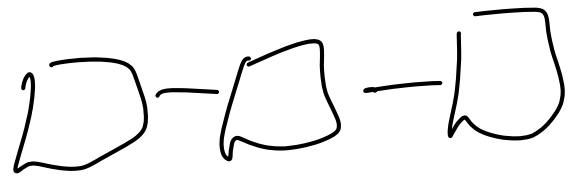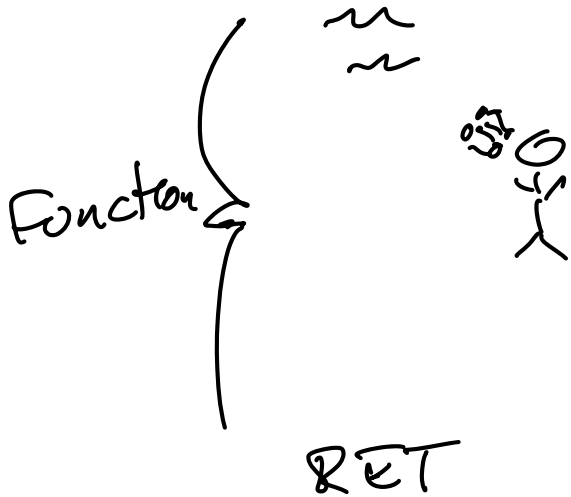
passing arguments   In registers (up to 6 ~~if~~)

non-MSFT: 1st 6 arguments in RDI, RSI, RDX, RCX,
            arg 7, ... → stack   R8, R9
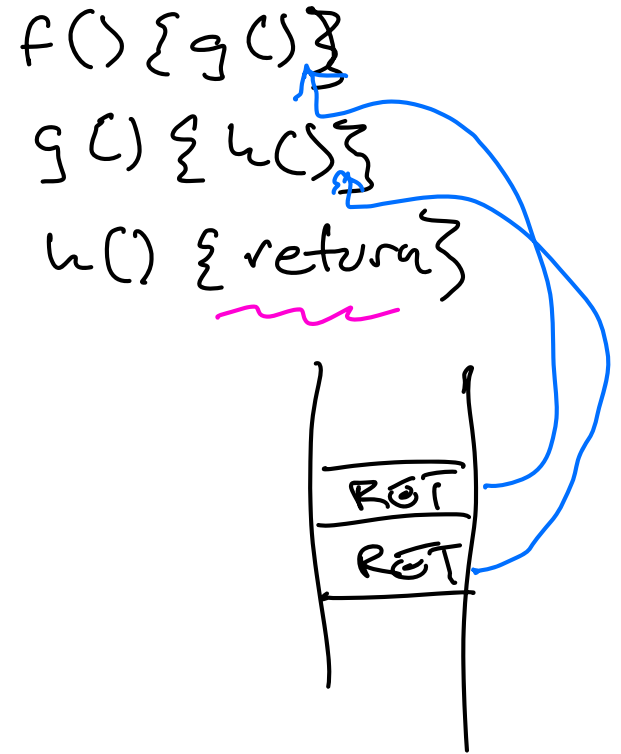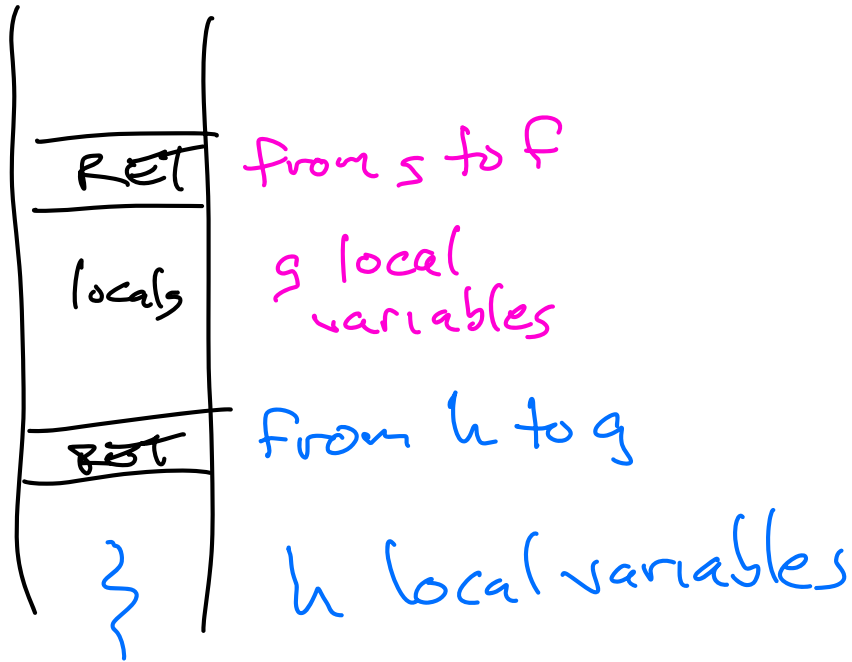
~~MSFT: RCX, RDX, R8, R9~~

machine code

Function {
         ~~
         ~~
              🏃

RET

CCC.   (a,b,c)
                      ;
              X = a

C:
void f(int a, int b){
   ;
   a ---> 1st arg.

local variables

RET
locals
RET

from s to f

g local variables

from h to g

h local variables

f() { g() }
g() { h() }
h() { return }

RET
RET

function prologue & epilogue

CALL f: ⟶ PUSH BP
COPY SP → BP
⟨
COPY BP → SP
POP BP
RET

```
RET
BP  •
  ⟨
RET
BP  •
  ⟨
```

BP →

BP ⟶ SP

RET
old BP
← a
← buf[3]
locals
← b
← x
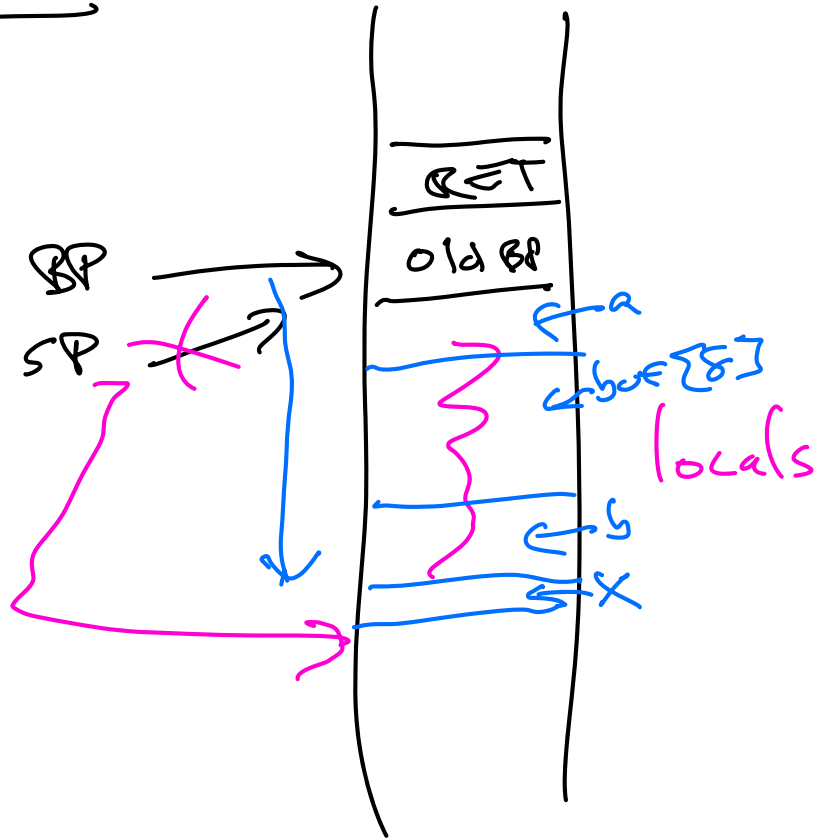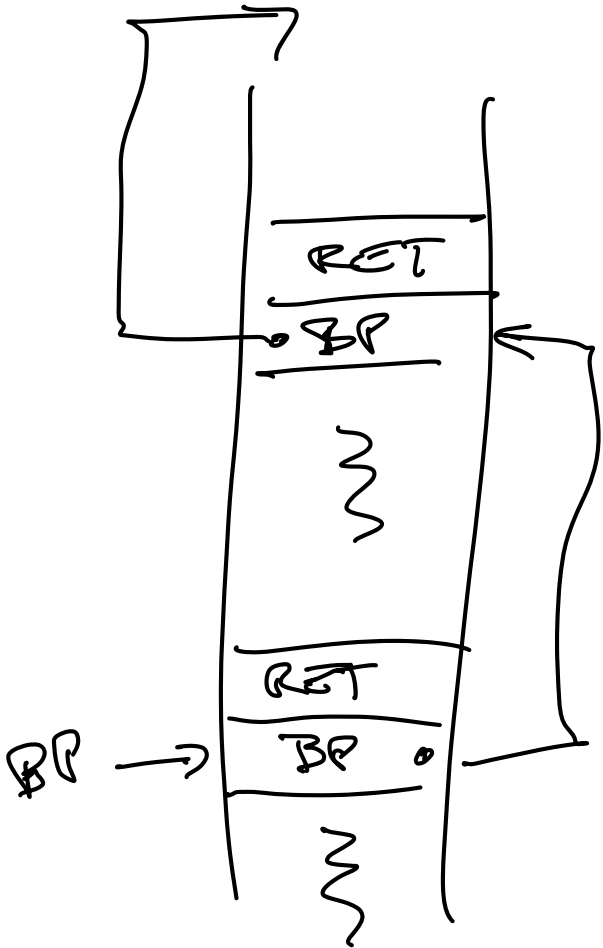
char * f (void) {
  char buf [4]
  --> buf
  return &buf
}

tokenize → argv

```
{
    char qbuf[16];
    sprintf(qbuf, "%d" ..
    argv[i] = qbuf
}
{
    int a;
    int b;
}
```

Pass argv → execvp

More about function calls

f.c   g.c

f()   g()

{
  ← register R8=1, R9=2,...

CALL g

PUSH R8   g()
:
POP R8

call-clobbered regs

call-preserved regs
        → RBX, R12..R15