# PintOS Project 2-1

2025-04-10

# Starting Project 2-1…

- Make a new branch named “project2-1” and do your work at that branch.
  - git branch project2-1
  - git checkout project2-1
- Since the project 2-1 and 1 is independent to each other, you can start regardless your implementation of project 1.
- However, the other projects are dependent to each other, **you should finish project 2-1 before start 2-2.**
- **Make commits frequently!**

# Pre-work

- **Read pintos manual**
  - 3. Project 2: User Programs
  - https://web.stanford.edu/class/cs140/projects/pintos/pintos_3.html#SEC32

- If you must change your simulator into qemu, please let us know, and write down at the bottom of the design document.

- Check your environment
  - `$ cd ~/uni{student_id}/project2-1/pintos/src/userprog`
  - `$ make`
  - `$ cd build`
  - `$ make check`
  - You can see that all test fails

# Goal for project 2-1

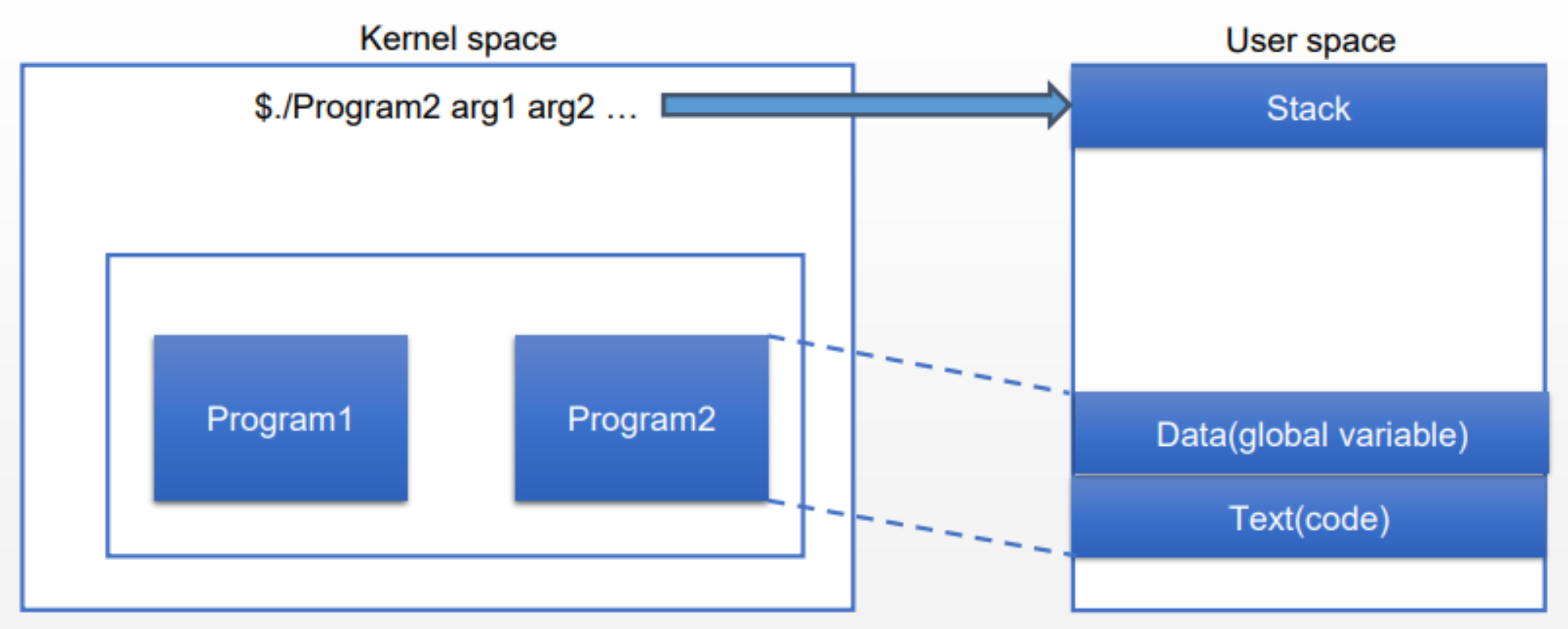
**1. Argument passing**

**2. System call implementation**

- halt, exit, create, open, close, write(partial)

# Concept for argument passing

Arguments : user input strings when executing a program

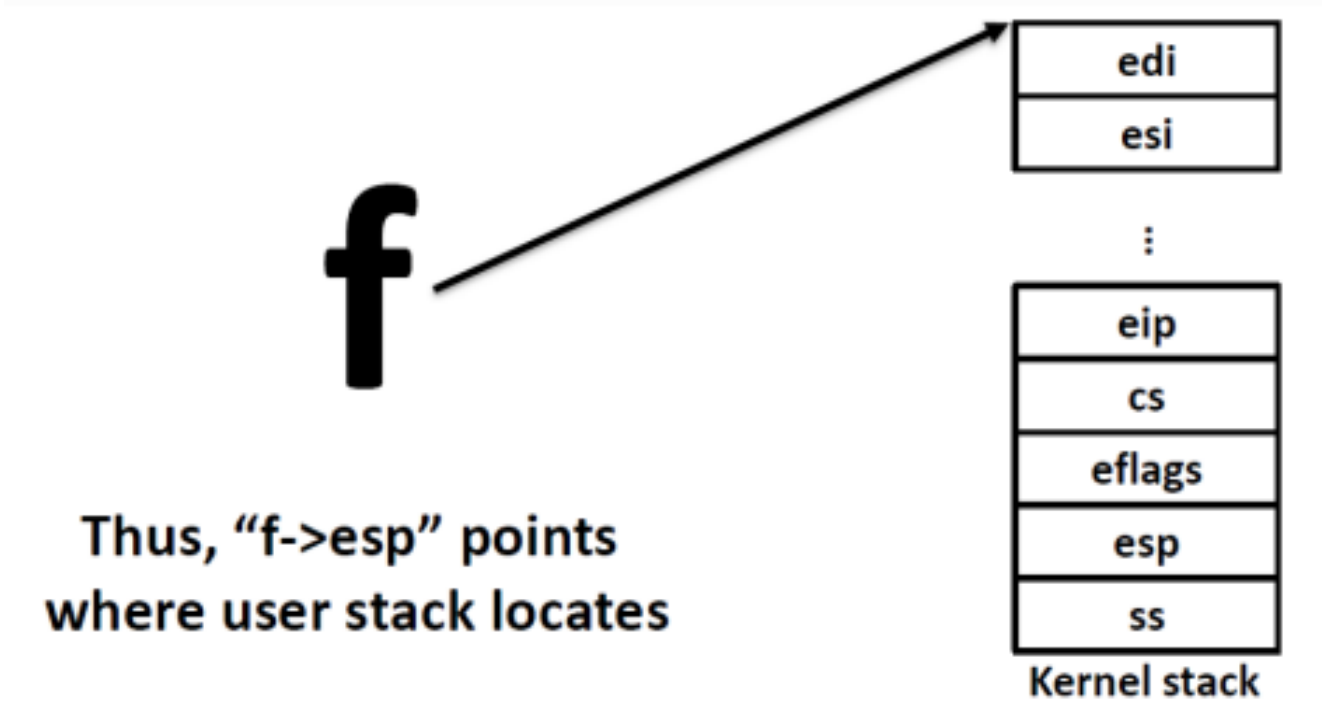


# Push arguments in the stack

- Example) `$/bin/ls -l foo bar`
  - Argument : `/bin/ls, -l, foo, bar`
- Each argument is separated with space
- You may want to use `strtok_r()` in `lib/string.c`
- For understanding this figure, you can read PintOS **manual 3.5, 80x86 calling convention**

# How to access kernel stack?

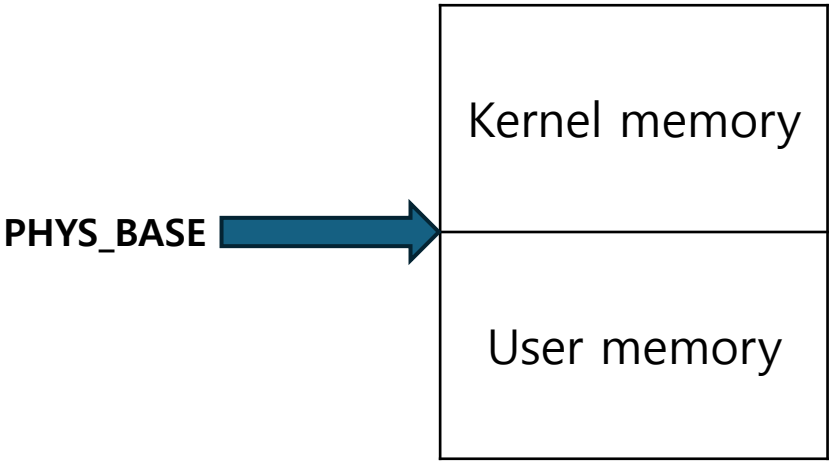
- `syscall_handler(struct intr_frame *f) //defined in syscall.c`
  - You have to implement this function



# User memory access

- On system calls, the kernel needs to access user memory

- Need to check whether the address is valid
  - It should be in user memory space(below PHYS_BASE)
  - It should be mapped in the page table
- Two ways to do it:
  - Do a full verification, then dereference
    You may want to use pagedir_get_page()
  - Only check if it is in user memory space
    You may need to edit page fault handler in userprog/exception.c



# File system for pintos

- No internal synchronization
- File size is fixed at creation time
- File data is allocated as a single extent
- No subdirectories
- File name are limited to 14 characters
- There is no file system repair tool anyway

- Detail information: manual 3.1.2

# Using the file system

- `$cd userprog`
- `$pintos-mkdisk filesys.dsk --filesys-size=2`

  -> it means that it creates 2MB disk names "filesys.dsk"

- `$pintos -- -f -q`

  -> formats the disk(-f) and exits as soon as the format is done(-q)

- `$ cd build && pintos -p ../../examples/echo -a echo -- -q`

  -> put the file "../../examples/echo" to the Pintos file system under the name "echo"
- `$ pintos -- -q run 'echo x'`

  -> Run the executable file "echo", passing argument "x"

# For file systems

- You will *need filesys/filesys.h, filesys/file.h* and *lib/syscall-nr.h* to implement system calls related to file systems

- You don't need to modify the system code (UNDER the *filesys* directory) for this project

# What to do

- You can refer
  - *userprog/process.c, syscall.c, lib/string.c*
  - pagedir.c : useful for implementing user memory access

- What syscalls should we implement?
  - halt() : to quit system running(shutdown)
  - exit() : to quit program running
  - create() : to create a file
  - open() : to open a file
  - close() : to close a file
  - write() : output to the console

# When you implement write()

- `int write (int fd, const void *buffer, unsigned size)`
  - **If fd==1, writes to console: call putbuf(buffer, size) and return size**
  - Else, write size bytes from buffer to the file opened with fd, return size
- You have to implement only highlighted part
- If you don’t implement this part, result of test case could not be printed
  - The rest part of write() will be implemented in project 2-2

# You have to pass…

```
pass tests/userprog/args-none
pass tests/userprog/args-single
pass tests/userprog/args-multiple
pass tests/userprog/args-many
pass tests/userprog/args-dbl-space
pass tests/userprog/sc-bad-sp
pass tests/userprog/sc-bad-arg
pass tests/userprog/sc-boundary
pass tests/userprog/sc-boundary-2
pass tests/userprog/halt
pass tests/userprog/exit
pass tests/userprog/create-normal
pass tests/userprog/create-empty
pass tests/userprog/create-null
pass tests/userprog/create-bad-ptr
pass tests/userprog/create-long
pass tests/userprog/create-exists
pass tests/userprog/create-bound
pass tests/userprog/open-normal
pass tests/userprog/open-missing
pass tests/userprog/open-boundary
pass tests/userprog/open-empty
pass tests/userprog/open-null
pass tests/userprog/open-bad-ptr
pass tests/userprog/open-twice
pass tests/userprog/close-normal
pass tests/userprog/close-twice
pass tests/userprog/close-stdin
pass tests/userprog/close-stdout
pass tests/userprog/close-bad-fd
```

# Grading

- Code (pintos grade) : 90%
- Design document : 10%
  - A1, A2, A3, A4, B1, B2, B5

# For single test grading..

- For example, you want to run “args-single” test
- For running this test, you have to use same command in make grade

```
[cs20121397@uni06 build]$ make grade
pintos -v -k -T 60 --bochs  --filesys-size=2 -p tests/userprog/multi-recurse -a multi-recurse -- -q  -f run 'multi-recurse 15' <
/dev/null 2> tests/userprog/multi-recurse.errors |tee tests/userprog/multi-recurse.output
```

- In here,

```
$ pintos -v -k -T 60 --bochs --filesys-size=2 -p tests/userprog/multi-recurse -a multi-
recurse -- -q -f run 'multi-recurse 15’
```

- For using this command, you have to run command in ***build directory***