Introduction to OS161

OS161 kit setup

Building the kernel

Running the kernel

Outline

- OS161 setup
- Understanding OS161
- Building OS161
- Running OS161

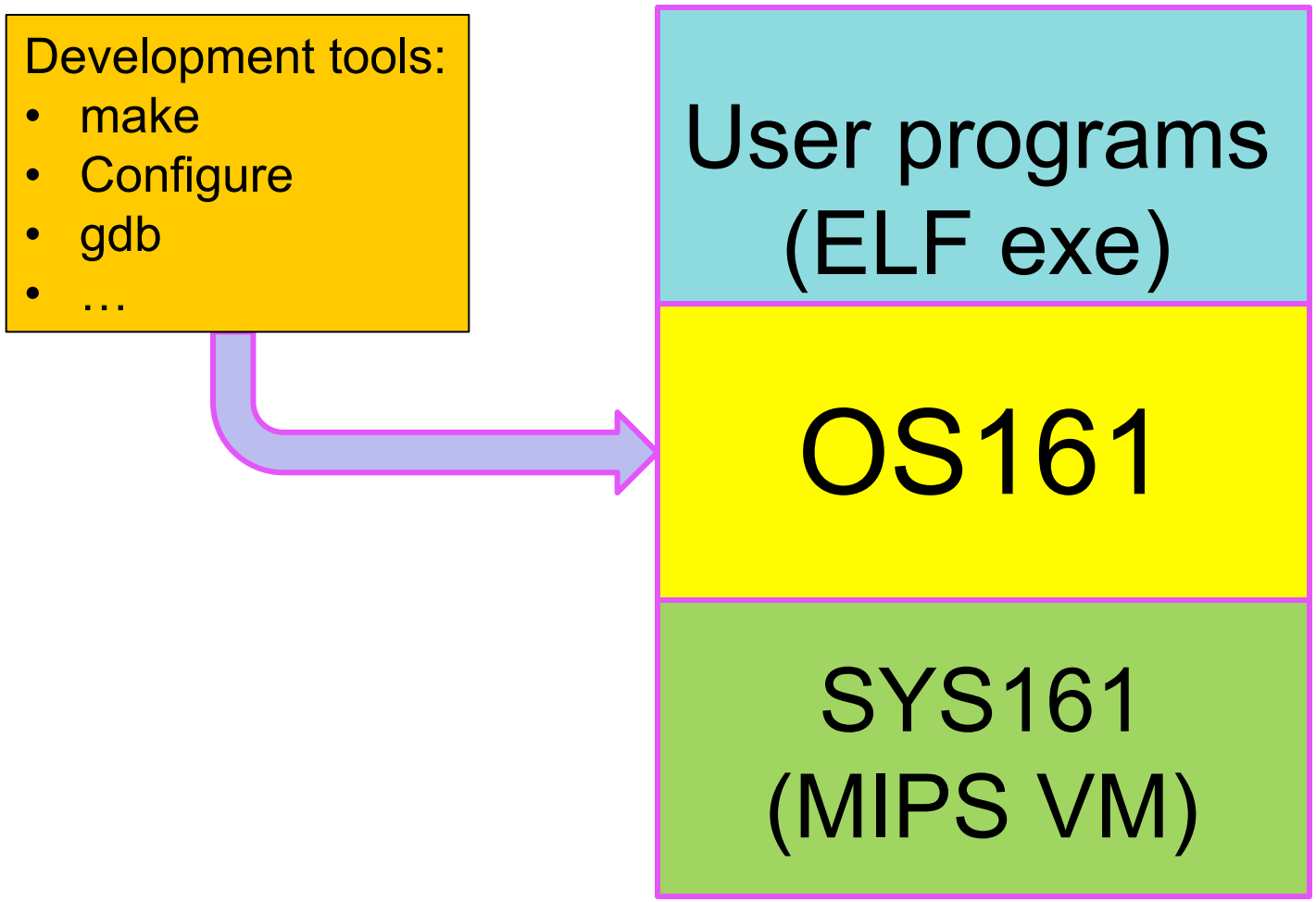
Os161

- teaching operating system (written in C) a simplified unix BSD-like OS
- runs on a simulator (MIPS VM).
- two supported branches
  - 1.x branch, uniprocessor kernel
  - 2.x branch, fully released in 2015, multiprocessor support and other
- includes both a kernel of conventional ("macrokernel") design and a simple userland, including a variety of test programs.

OS161 framework

OS161 includes

- the sources of the operating system (kernel), to be used for
  - code browsing
  - designing, implementing new/missing features
  - running and debugging
- a toolchain for
  - cross compiling (OS161 kernel for a MIPS processor)
  - running the kernel on top of a machine simulator called sys161
  - other tasks…


Development tools:
• make
• Configure
• gdb
• …
User programs
(ELF exe)
OS161
SYS161
(MIPS VM)

OS161 support

- The base OS161 system **provides** low-level trap/interrupt, device drivers, in-kernel threads, a baseline scheduler, a minimal virtual memory system, a simple file system
- Other things (not included) **have to be implemented**:
  - Locks.
  - System calls.
  - Virtual memory. The "dumbvm" shipped with OS161 is good enough for bootstrapping and doing the early assignments. It never reuses memory and cannot support large processes or malloc.
  - File system.
- Many other things can be added to OS161

Understanding OS161
(ASST0: first lab)

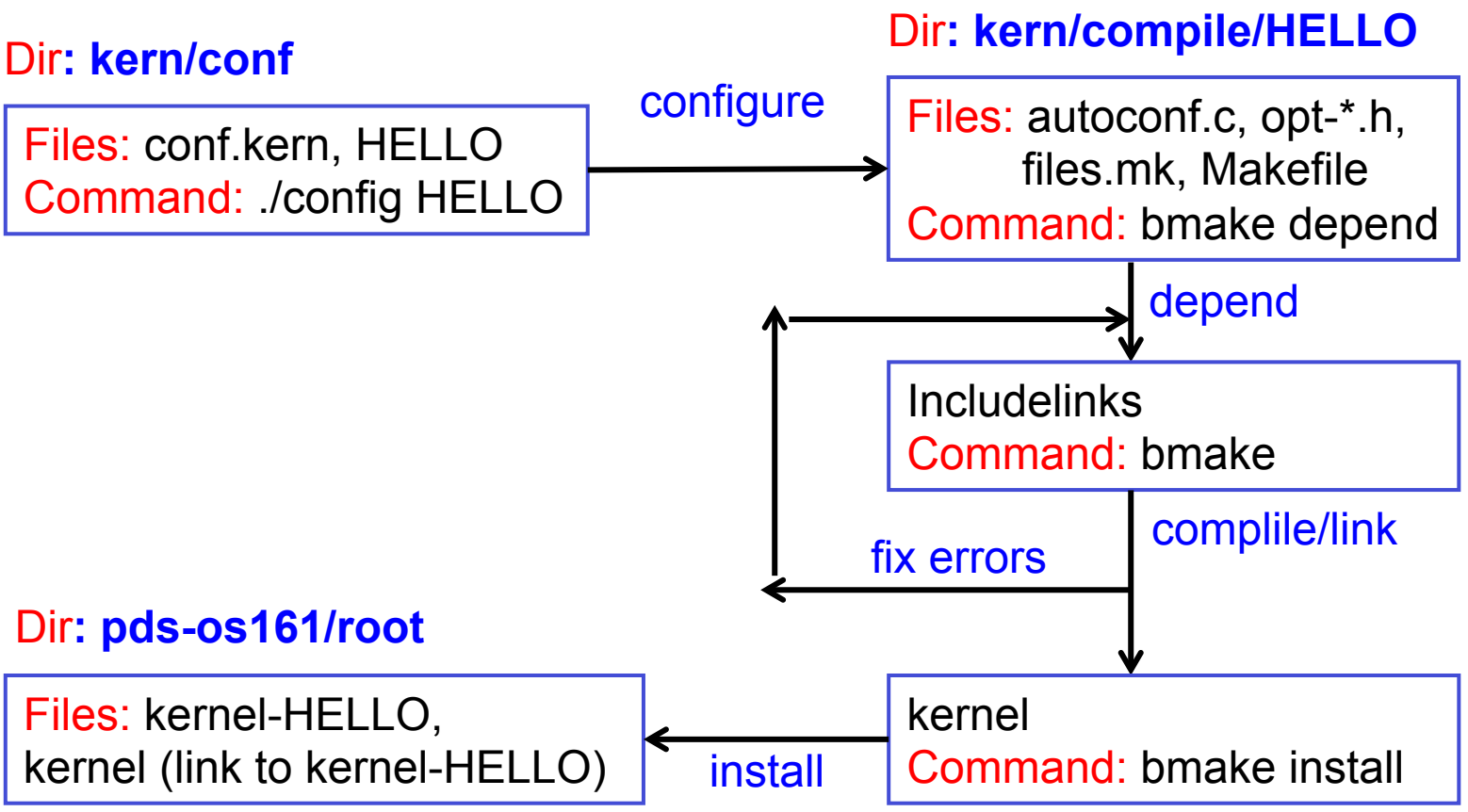
- Set up OS161 development environment.
- Understand the source code structure of OS161.
- Navigate the OS/161 sources to determine where and how things are done.
- Be able to modify, build (configure, bmake) and run OS/161 kernel.
- Use GDB.

Working in OS161

Directory tree (PdS Ubuntu 14.04 virtual machine):

- /home/pds: pds user directory
- os161_doc: documentation (with browsable code)
- pds-os161/root (full path: /home/pds/pds-os161/root): run/execution
  - pds-os161/root/testscripts: user program execs (from userland) to be called within os161 test menu.
- os161 (full path: /home/pds/os161): tools and os161 source/build
  - os161/tools: tools fpr compilation, make(build), debug (eg. mips-harvard.os161-gcc)
  - os161/os161-base-2.0.2: building kernel and user programs
  - os161/os161-base-2.0.2/userland: user source programs (e.g. test)
  - os161/os161-base-2.0.2/kern: kernel source
  - os161/os161-base-2.0.2/kern/conf: kernel configuration
  - os161/os161-base-2.0.2/kern/compile: kernel compilation/build

Making (building) OS161 new release: HELLO



Making (building) OS161

Code browsing/understanding

- Edit .c/.h files in os161/os161-base-2.0.2/kern
- Use browsable code from os161_doc/os161/html/index.html

Kernel configuration/options

- os161/os161-base-2.0.2/kern/conf/conf.kern: definition of options and list of files
- 4 kernel configurations already available: DUMBVM, DUMBVM-OPT, GENERIC, GENERIC-OPT (they include conf.kern).
- To generate a new configuration, copy and modify: e.g. HELLO (new configuration) copied from GENERIC and modified
- COMMAND (in os161/os161-base-2.0.2/kern/conf)
  - ./config HELLO
  - Generates os161/os161-base-2.0.2/kern/compile/HELLO

Making (building) OS161

Compilation/make

- In os161/os161-base-2.0.2/kern/compile/HELLO (or equivalent directory)
- Make dependencies: scan C files and generate rules to (automatically) recompile a given source C file (generate the object file) if a .h is modified
  - bmake depend
- Compile (build executable: e.g. kernel-HELLO)
  - bmake
  - if compilation errors, correct code and rerun
- Install (copy) executable in pds-os161/root
  - bmake install
  - Copies kernel-HELLO (or other) and generates symbolic link “kernel”

Running/debugging

Work in pds-os161/root

MIPS virtual machine (sys161) configured in sys161.conf

- One important line to be properly edited
  - mainboard ramsize=1024K cpus=1
- Running (bootstrap) kernel on mips machine
  - sys161 kernel (without debugger support)
  - sys161 –w kernel (with debugger support: waiting for debugger connection on socket)
  - … or other

OS161 kernel

Kernel main (kmain)

- os161/os161-base-2.0.2/kern/main
- main.c, menu.c

```
void kmain(char *arguments) {
  boot();
  menu(arguments);
  /* Should not get here */
}
```

- Basic support (partial): threads, memory management, system calls, semaphores, running user executable (ELF format)