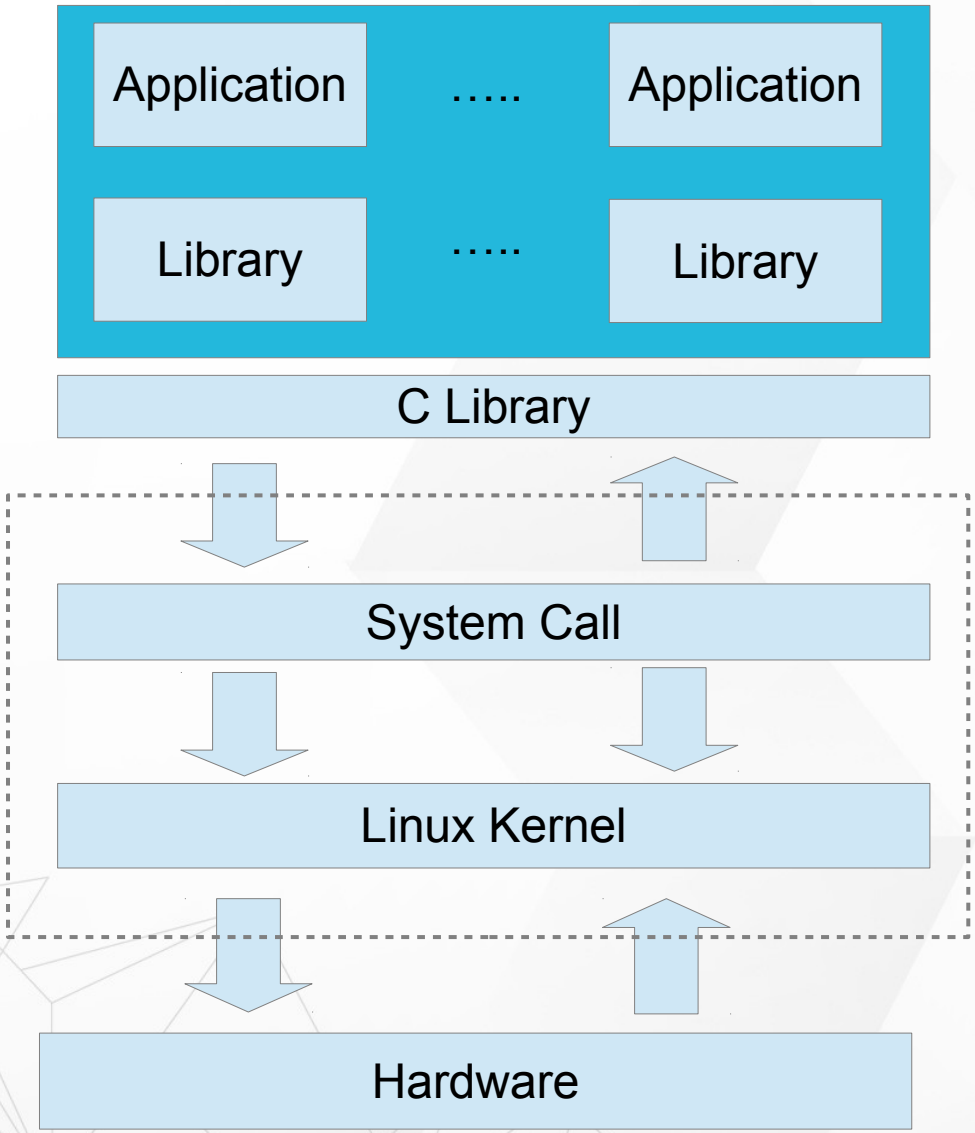
Linux Kernel

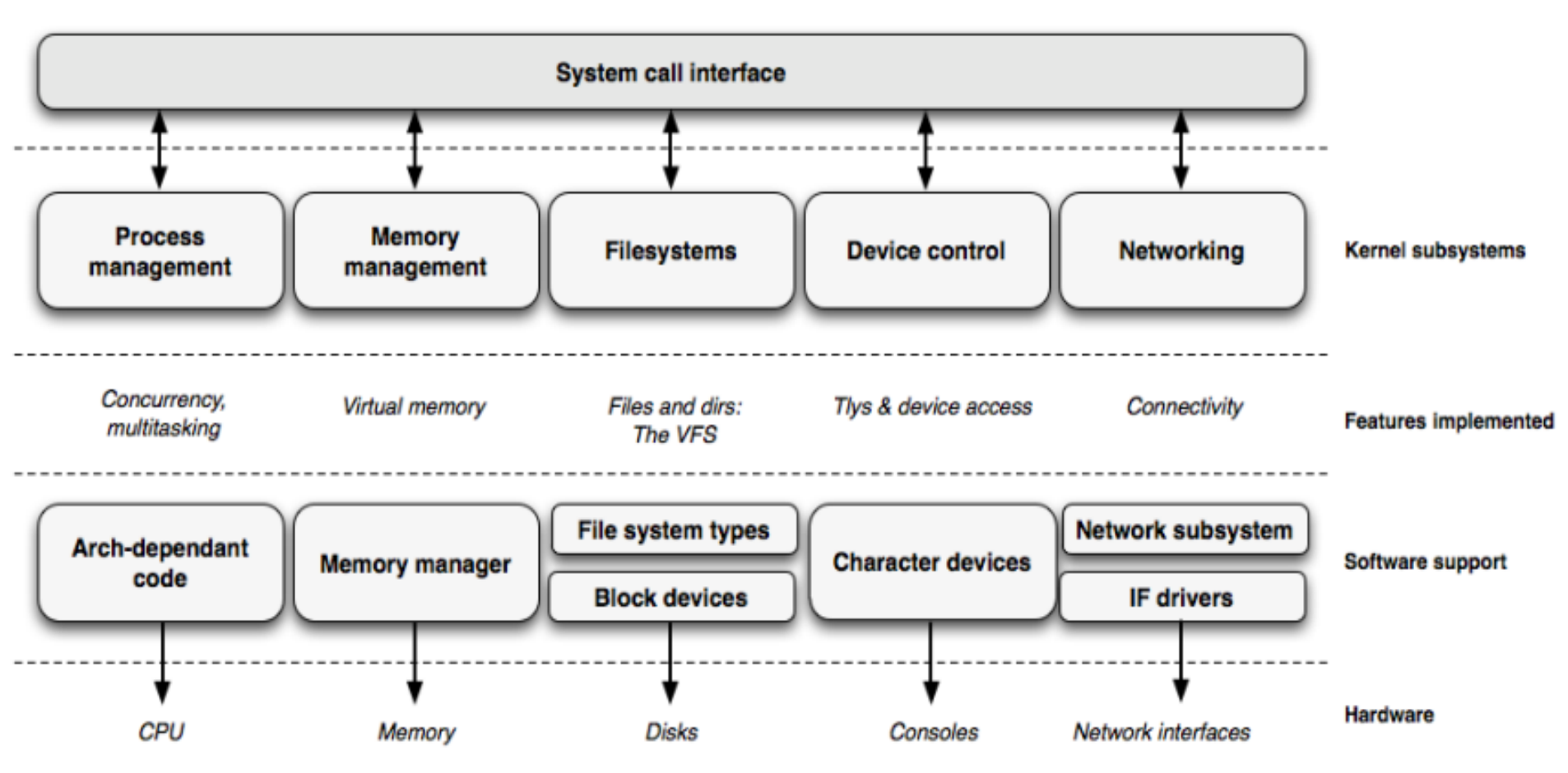
Linux kernel key features

- Portability and hardware support
- Scalability
- Compliance to standards and interoperability
- Exhaustive networking support
- Stability and reliability
- Modularity
- Easy to program.

Linux kernel in the system



Linux kernel



Kernel Source

- https://www.kernel.org/
- Many chip vendors
- kernel sub-communities
  - Architecture communities
    - ARM, MIPS, PowerPC ...
  - device drivers communities
    - I2C, SPI, USB, PCI, network ...

Programming language

- Implemented in C like all Unix systems
- A little Assembly is used too
- **No C++ used**
- **No C library**
- **No floating point** computation
- Kernel code has to supply its own library implementations
  - X : printf(), memset(), malloc(),...
  - O: printk(), memset(), kmalloc()

Linux sources important folder

- Kernel Image
  - arch/<ARCH>/boot/
  - Arch/arm64/boot
- DTS
  - arch/<ARCH>/boot/dts
  - Arch/arm64/boot/dts/rockchip/
- driver/
- Documentation/

Kernel Basic Command

Basic Build Command

- $ make
  → Build all

- $ make dtbs
  → Build Device-tree only

- $ make modules
  → Build kernel modules only

- $ make modules_install
  → Install all modules to INSTALL_MOD_PATH

Basic Build Command

- $ make modules_install
  → Install all modules to INSTALL_MOD_PATH
- $ make mrproper
  → Remove all generated files (.config)
- $ make clean
- $ make distclean
  → Remove editor backup and patch reject files

Make help

$ make help

```
acs5k_defconfig           - Build for acs5k
acs5k_tiny_defconfig      - Build for acs5k_tiny
afeb9260_defconfig        - Build for afeb9260
ag5evm_defconfig          - Build for ag5evm
am200epdkit_defconfig     - Build for am200epdkit
ap4evb_defconfig          - Build for ap4evb
armadillo800eva_defconfig - Build for armadillo800eva
assabet_defconfig         - BuiOther generic targets:
at91_dt_defconfig         - Bui   all              - Build all targets marked with [*]
at91rm9200_defconfig      - Bui* vmlinux           - Build the bare kernel
at91sam9260_defconfig     - Bui* modules           - Build all modules
at91sam9261_defconfig     - Bui   modules_install  - Install all modules to INSTALL_MOD_PATH (default: /)
at91sam9263_defconfig     - Bui   firmware_install- Install all firmware to INSTALL_FW_PATH
at91sam9g20_defconfig     - Bui                      (default: $(INSTALL_MOD_PATH)/lib/firmware)
at91sam9g45_defconfig     - Bui   dir/             - Build all files in dir and below
at91sam9rl_defconfig      - Bui   dir/file.[oisS]  - Build specified target only
at91x40_defconfig         - Bui   dir/file.lst     - Build specified mixed source/assembly target only
badge4_defconfig          - Bui                      (requires a recent binutils and recent build (System.map))
bcmring_defconfig         - Bui   dir/file.ko      - Build module including final link
bonito_defconfig          - Bui   modules_prepare  - Set up for building external modules
cam60_defconfig           - Bui   tags/TAGS        - Generate tags file for editors
cerfcube_defconfig        - Bui   cscope           - Generate cscope index
cm_x2xx_defconfig         - Bui   gtags            - Generate GNU GLOBAL index
cm_x300_defconfig         - Bui   kernelrelease    - Output the release version string
cns3420vb_defconfig       - Bui   kernelversion    - Output the version stored in Makefile
colibri_pxa270_defconfig  - Bui   headers_install - Install sanitised kernel headers to INSTALL_HDR_PATH
colibri_pxa300_defconfig  - Bui                      (default: /home/xlloss/work/tiny-4412/build/linux_3.5.0_tiny4412/usr)
collie_defconfig          - Bui
corgi_defconfig           - Bui
cpu9260_defconfig         - Build for cpu9260
cpu9g20_defconfig         - Build for cpu9g20
```

How to Build Linux Kernel

Specifying cross-compilation

- make **ARCH=arm64 CROSS_COMPILE=arm-linux-** …

  → export ARCH=arm64

  → export CROSS_COMPILE=aarch64-linux-gnu-

- Add above setting to script

  → source $PATH/**set_toolchain.sh**

set_toolchain.sh

```
export PATH=$PATH:${TOOLCHAIN}/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin
# Toolchain path add to environment variable

export ARCH=arm64
# Set SOC architecture type

export CROSS_COMPILE=aarch64-linux-gnu-
#Set compile prefix name

export KERNELDIR=${KERNEL_PATH}/rockchip-rk3399-nanopi-m4
#Set Linux kernel source path
```

Predefined configuration files

- **Default configuration**
  - arch/<arch>/configs/
- **make nanopi4_linux_defconfig**
- To create your own default configuration file

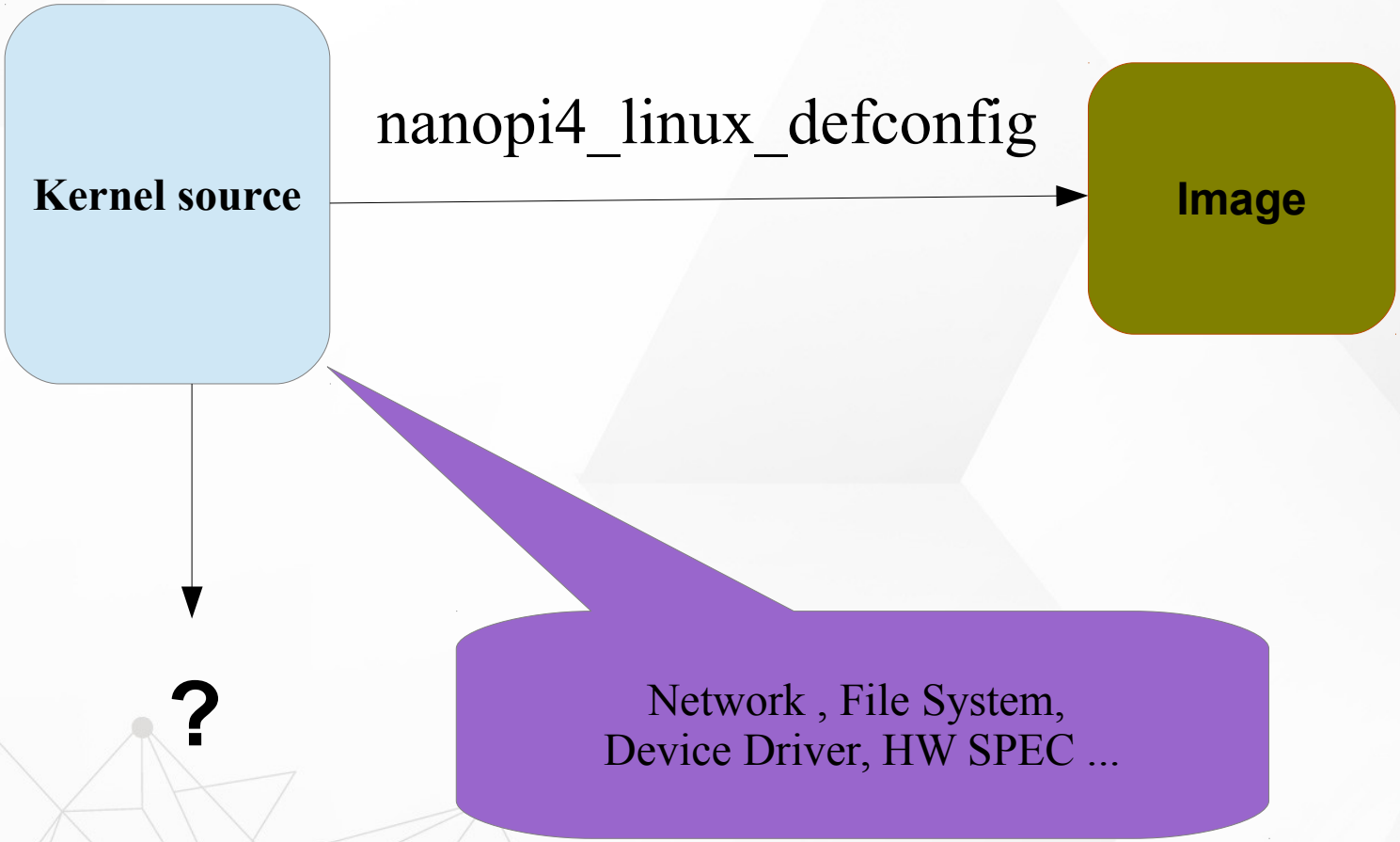
  → make savedefconfig, to create a minimal configuration file

  → mv defconfig arch/arm64/configs/myown_defconfig

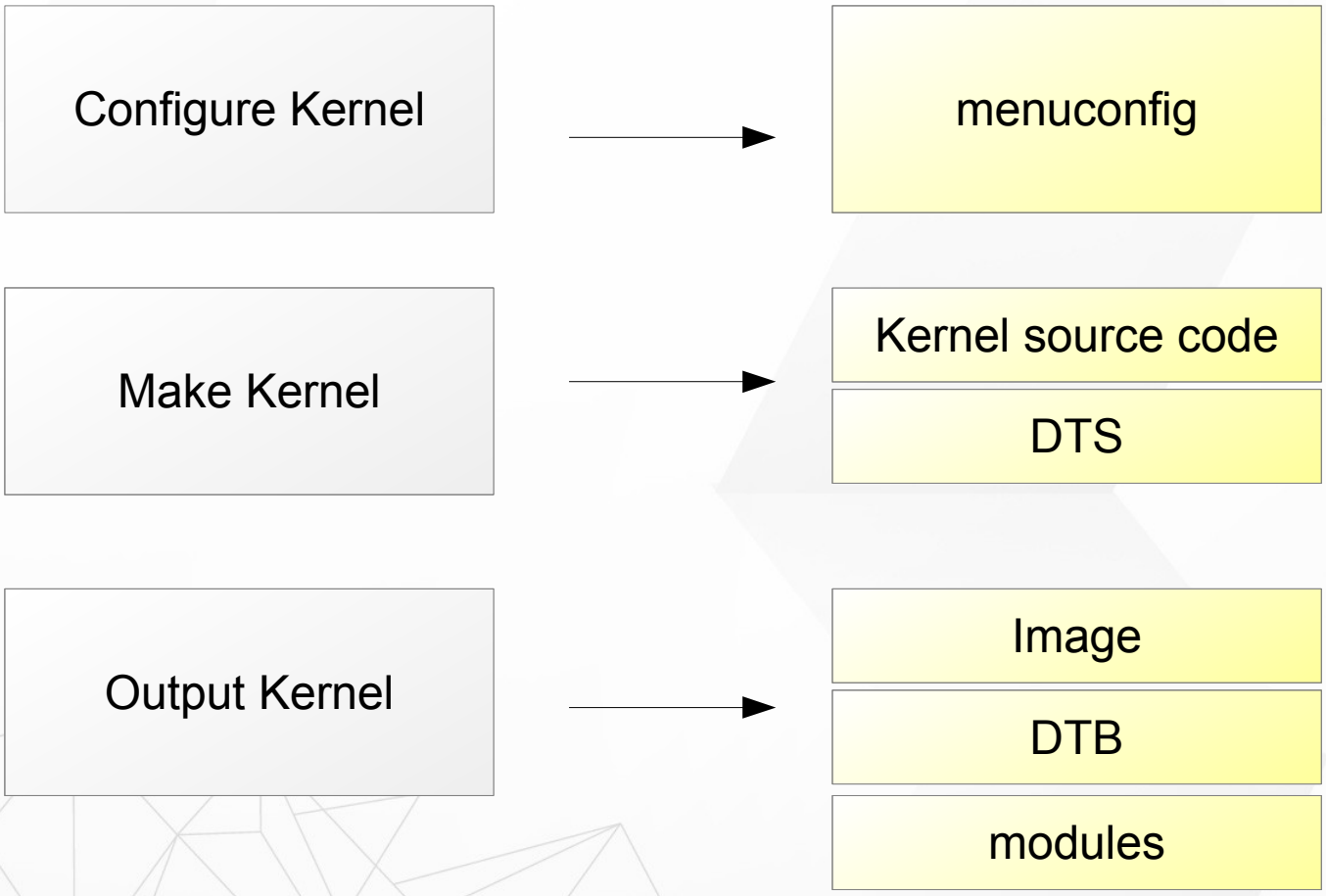
Kernel compilation

- Build kernel Image → **make -j4**
  - To run multiple jobs in parallel if you have multiple CPU cores
- Generates Image
  - arch/arm64/boot/Image
    → **Image** for ARM64,
  - arch/arm64/boot/dts/rockchip/
    → DTB : rk3399-nanopi4-rev01.dtb
    → DTB : rk3399-nanopi4-rev21.dtb

Kernel configuration



Kernel configuration



How to Select Feature in Kernel

Kernel configuration

- The kernel configuration and build system is based on multiple **Makefiles**
- The configuration is stored in the .config file at the root of kernel sources
- As options have dependencies, typically never edited by hand, but through graphical or text interfaces
  - make menuconfig → Text
  - make xconfig → graphical

Kernel configuration

```
  4096 Apr 20 11:56 .
  4096 Mar 31 08:42 ..
  4096 Mar 31 08:41 android
  4096 Mar 31 08:41 arch
   419 Mar 31 08:41 backported-features
  4096 Mar 31 08:41 block
   459 Mar 31 08:41 build.config.cuttlefish.aarch64
   457 Mar 31 08:41 build.config.cuttlefish.x86_64
   296 Mar 31 08:41 build.config.goldfish.arm
   303 Mar 31 08:41 build.config.goldfish.arm64
   277 Mar 31 08:41 build.config.goldfish.mips
   279 Mar 31 08:41 build.config.goldfish.mips64
   298 Mar 31 08:41 build.config.goldfish.x86
   303 Mar 31 08:41 build.config.goldfish.x86_64
  4096 Mar 31 08:41 certs
     9 Mar 31 08:41 .checkpatch.conf
154048 Apr 20 11:56 .config
```

.config

```
cuttlefish_defconfig
defconfig
lsk_defconfig
nanopi4_linux_defconfig
px30_linux_defconfig
px30_linux_robot_defconfig
ranchu64_defconfig
rk1808_linux_defconfig
rk1808_x4_linux_defconfig
rk3308_linux_defconfig
rk3326_linux_defconfig
rk3326_linux_robot_defconfig
rk3399pro_npu_defconfig
rk3399pro_npu_pcie_defconfig
rockchip_cros_defconfig
rockchip_defconfig
rockchip_linux_defconfig
```

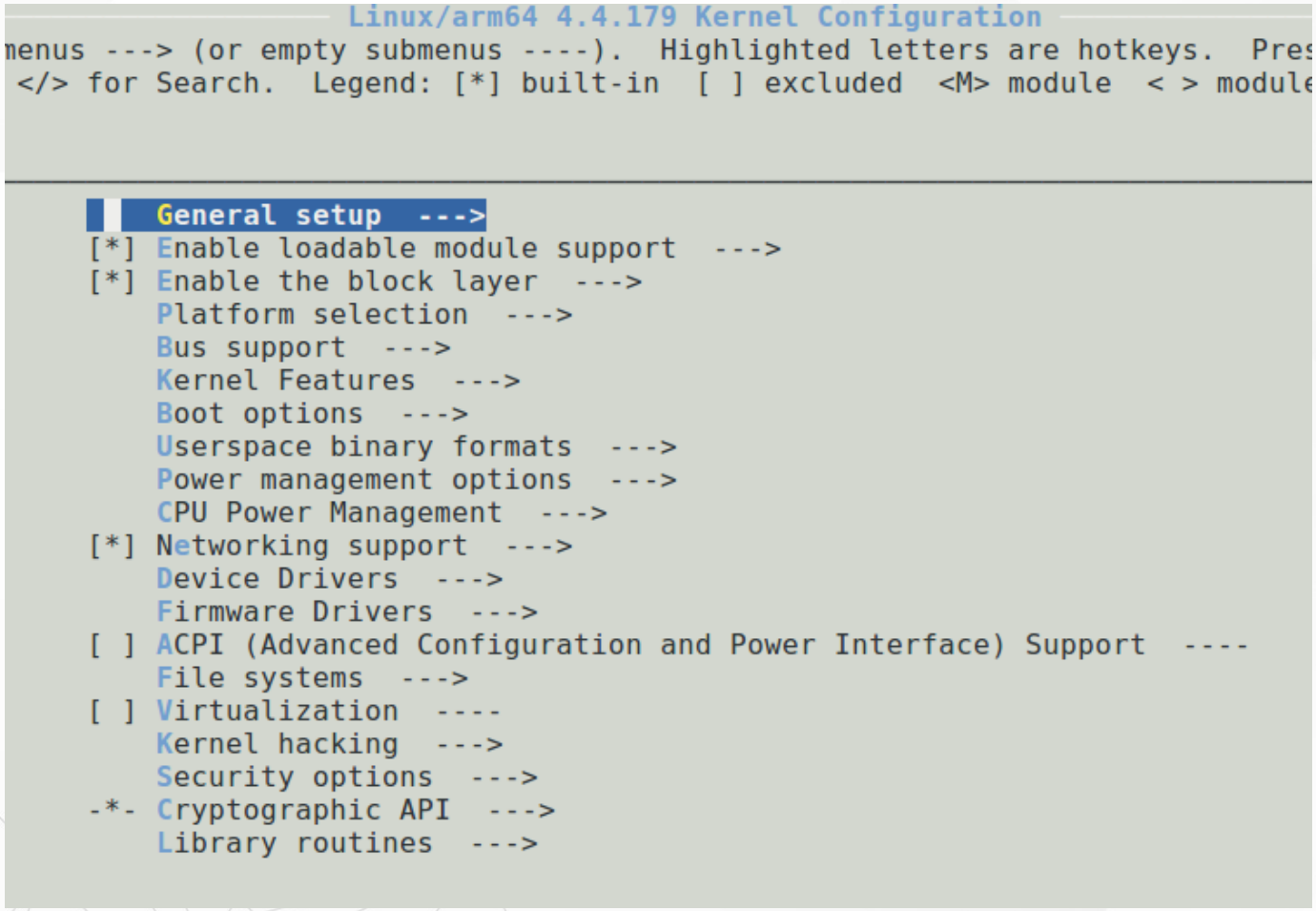
${KERNEL}/arch/arm64/configs/

nanopi4_linux_defconfig

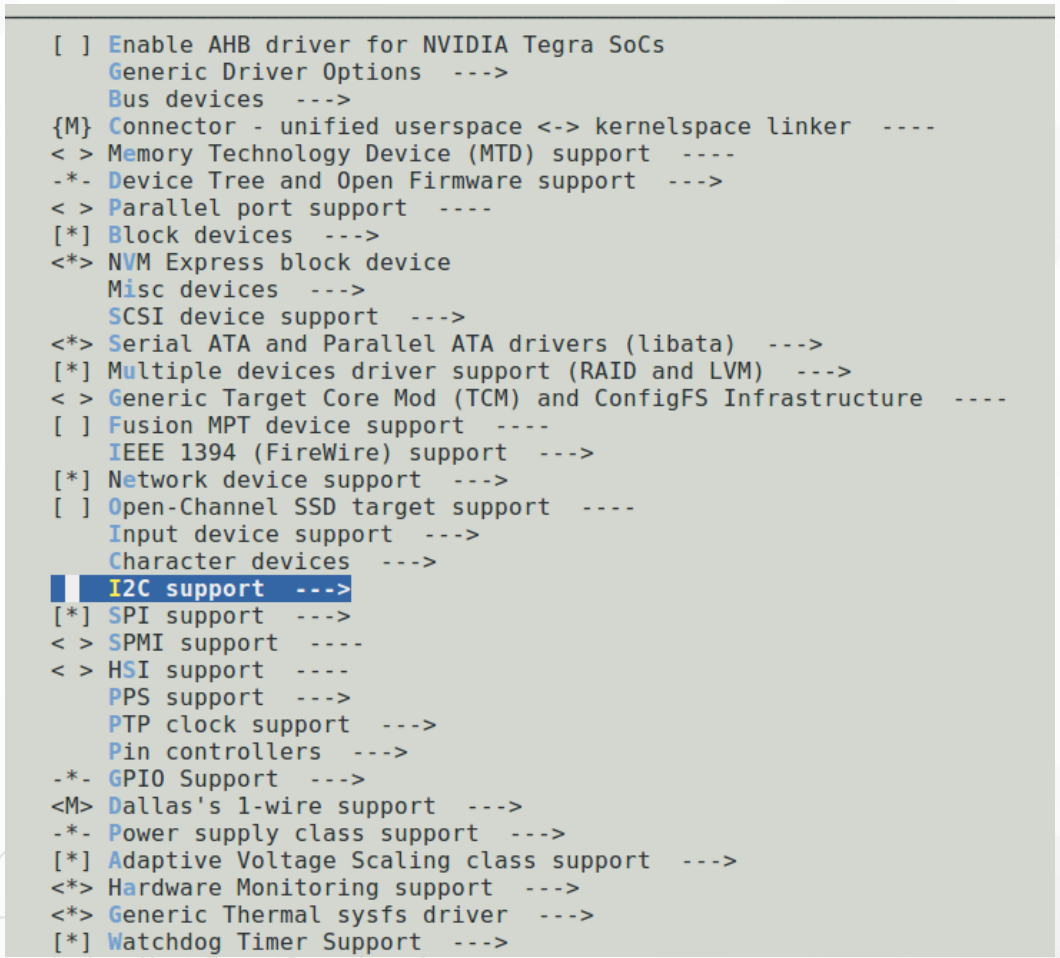
Kernel or module?

- The kernel image is a single file, resulting from the linking of all object files that correspond to features enabled in the configuration
- Some features (device drivers, file-system, etc.) can however be compiled as modules

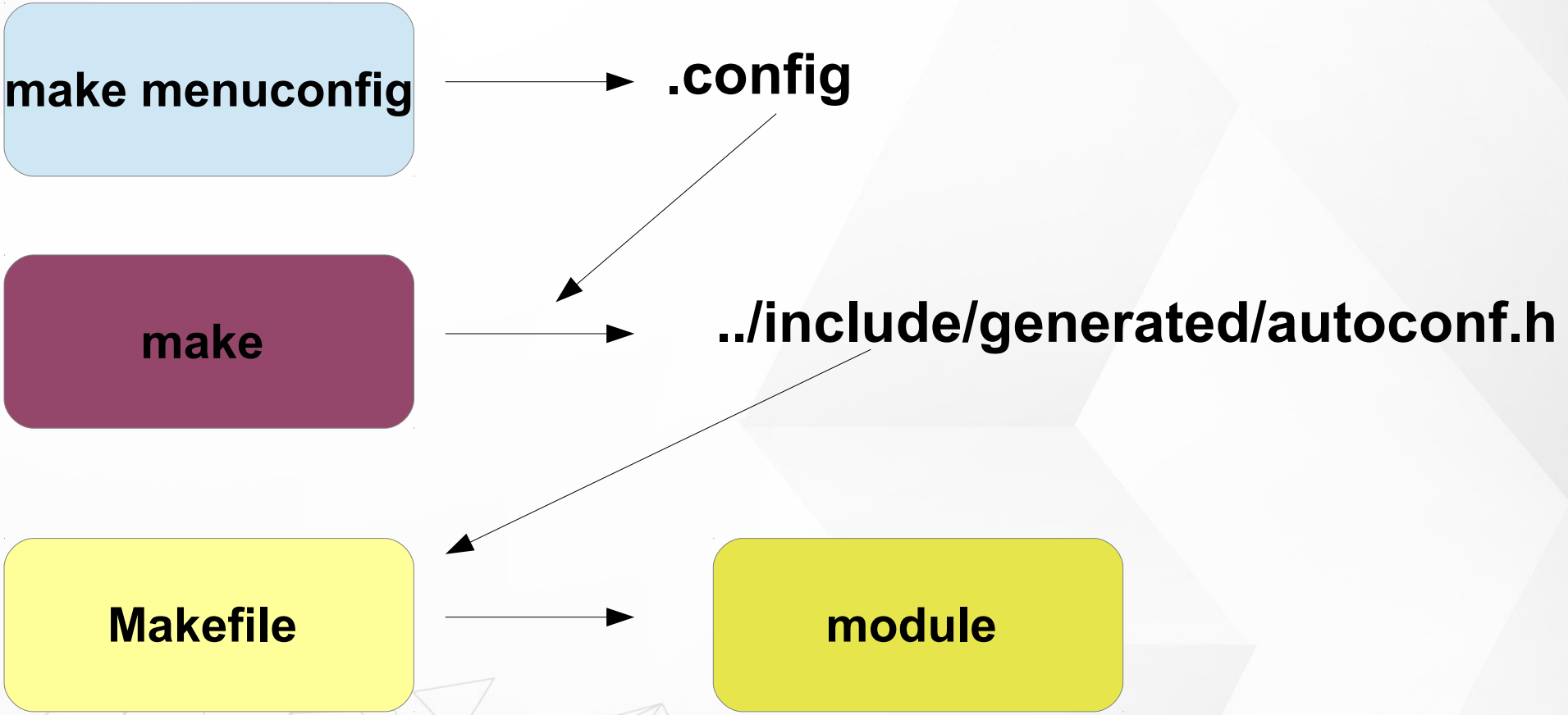
menuconfig



Kernel Configuration Options



Configuration



Corresponding .config File Excerpt

```
#
# I2C system bus drivers (mostly embedded / system-on-chip)
#
# CONFIG_I2C_CADENCE is not set
# CONFIG_I2C_CBUS_GPIO is not set
# CONFIG_I2C_DESIGNWARE_PLATFORM is not set
# CONFIG_I2C_DESIGNWARE_PCI is not set
# CONFIG_I2C_EMEV2 is not set
CONFIG_I2C_GPIO=m
# CONFIG_I2C_NOMADIK is not set
# CONFIG_I2C_OCORES is not set
# CONFIG_I2C_PCA_PLATFORM is not set
# CONFIG_I2C_PXA_PCI is not set
CONFIG_I2C_RK3X=y
# CONFIG_I2C_SIMTEC is not set
# CONFIG_I2C_XILINX is not set
```

.config

$(KERNEL_PATH)/drivers/i2c/buses/
Makefile

```
obj-$(CONFIG_I2C_PXA_PCI)       += i2c-pxa-pci.o
obj-$(CONFIG_I2C_QUP)           += i2c-qup.o
obj-$(CONFIG_I2C_RIIC)          += i2c-riic.o
obj-$(CONFIG_I2C_RK3X)          += i2c-rk3x.o
obj-$(CONFIG_I2C_S3C2410)       += i2c-s3c2410.o
obj-$(CONFIG_I2C_SH7760)        += i2c-sh7760.o
obj-$(CONFIG_I2C_SH_MOBILE)     += i2c-sh_mobile.o
obj-$(CONFIG_I2C_SIMTEC)        += i2c-simtec.o
obj-$(CONFIG_I2C_SIRF)          += i2c-sirf.o
```

Linux Kernel Booting

- Bootloader run kernel with parameters
  - x0 = DTB (Device Tree Blob)
  - r1 = NULL
  - r2 = NULL

Kernel Startup Entry Point

arch/arm64/kernel/head.S

```
ENTRY(stext)
    bl    preserve_boot_args
    bl    el2_setup            // Drop to EL1, w20=cpu_boot_mode
    adrp    x24, __PHYS_OFFSET
    and x23, x24, MIN_KIMG_ALIGN - 1    // KASLR offset, defaults to 0
    bl    set_cpu_boot_mode_flag
    bl    __create_page_tables        // x25=TTBR0, x26=TTBR1
    /*
     * The following calls CPU setup code, see arch/arm64/mm/proc.S for
     * details.
     * On return, the CPU will be ready for the MMU to be turned on and
     * the TCR will have been set.
     */
    bl    __cpu_setup            // initialise processor
    adr_l    x27, __primary_switch        // address to jump to after
                                // MMU has been enabled
    b    __enable_mmu
ENDPROC(stext)
```

Linux Kernel Booting