ktest.pl – tutorial
(Embedded Edition)

Steven Rostedt
srostedt@redhat.com
rostedt@goodmis.org

4096R/5A56DE73
5ED9 A48F C54C 0A22 D1D0
804C EBC2 6CDB 5A56 DE73
What is ktest.pl?

• A script written in perl
  – But you do not need to know perl!
• Written to build, install, boot and test kernels remotely
• Tests sets of commits in git
• normal building of kernel (also randconfig)
• bisect (git bisect and config bisect)
• make_min_config
Where is it?

• From Linux 2.6.38
  – tools/testing/ktest
• ktest.pl
  – The script to run
• samples.conf
  – Explains all config options that ktest.pl uses
Requirements

- Two machines
  - host
  - target (may be external or virtual machine)
- Host be able to remotely power cycle target
- Host be able to read target's console
- Source and Build directories must be separate
- Some tests require source to be a git repo
  - May add quilt support
My Setup

Thinkpad T60

Ethernet hub

snowball board

Digital Loggers
Web Power Switch
Digital Loggers
Power Cycle

- Cycle box connected to outlet 1 “outlet?1”

```
wget --no-proxy -O /dev/null -q --auth-no-challenge 'http://admin:admin@power/outlet?1=CCL'
```
Digital Loggers

Turn off

- Power off box connected to outlet 1
  “outlet?1”

wget --no-proxy -O /dev/null -q --auth-no-challenge 'http://admin:admin@power/outlet?1=OFF'
Digital Loggers
Turn on

- Power on box connected to outlet 1
  "outlet?1"

wget --no-proxy -O /dev/null -q --auth-no-challenge 'http://admin:admin@power/outlet?1=ON'
My Setup

- Thinkpad T60
- Ethernet hub
- snowball board
- Digital Loggers
- Web Power Switch
My Setup

Thinkpad T60

Ethernet hub

snowball board

My Thumb
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.13.0 netmask 255.255.255.0 {
    range dynamic-bootp 192.168.13.100 192.168.13.190;
    option broadcast-address 192.168.13.255;
    next-server 192.168.13.1;
    option subnet-mask 255.255.255.0;
    filename "snowball-image";
}
My Setup
(/etc/dhcpd/dhcpd.conf)

default-lease-time 600;
max-lease-time 7200;

subnet 192.168.13.0 netmask 255.255.255.0 {
  range dynamic-bootp 192.168.13.100 192.168.13.190;
  option broadcast-address 192.168.13.255;
  next-server 192.168.13.1;
  option subnet-mask 255.255.255.0 255.255.255.0;
  filename "snowball-image";
}

My Setup
(/etc/xinetd.d/tftp)

service tftp
{
    socket_type = dgram
    protocol = udp
    wait = yes
    user = root
    server = /usr/sbin/in.tftpd
    server_args = -s /var/lib/tftpboot
    disable = no
    per_source = 11
    cps = 100 2
    flags = IPv4
}
My Setup
(Problems with tftp?)

$ tftp localhost
tftp> get snowball-image
Error code 0: Permission denied
tftp>
Turn off selinux

# setenforce 0
My Setup
(snowball: printenv)

loadaddr=0x00100000
console=ttyAMA2,115200n8
memargs256=mem=96M@0 mem_modem=32M@96M mem=32M@128M
hwmem=22M@160M pmem_hwb=42M@182M mem_mali=32@224M
memargs512=mem=96M@0 mem_modem=32M@96M hwmem=32M@128M
mem=64M@160M mem_mali=32M@224M pmem_hwb=128M@256M mem=128M@384M
memargs1024=mem=128M@0 mali.mali_mem=32M@128M hwmem=168M@M160M
mem=48M@328M mem_issw=1M@383M mem=640M@384M
memargs=setenv bootargs ${bootargs} ${memargs1024}
emmcload=fat load mmc 0:2 ${loadaddr} /uImage
mmcload=fat load mmc 1:1 ${loadaddr} /uImage
commonargs=setenv bootargs console=${console} ip=dhcp vmalloc=256M
emmcargs=setenv bootargs ${bootargs} root=/dev/mmcblk0p3 rootwait
emmcboot=echo Booting from eMMC ...; run commonargs emmcargs memargs; bootm
    ${loadaddr}
mmcargs=setenv bootargs ${bootargs} root=/dev/mmcblk1p2 rootwait
mmcboot=echo Booting from external MMC ...; run commonargs mmcargs memargs; bootm
    ${loadaddr}
bootcmd=mmc rescan 0; mmc rescan 1; setenv ethaddr 0e:5e:d3:bf:97:4a; run commonargs
emmcargs memargs; bootp; bootm 0x00100000
Reading Console

• ttywatch
  - /etc/ttywatch.conf
    --name USB0 --port /dev/ttyUSB0 --bps 115200 --ipport 3001
  - telnet localhost 3001
  - nc localhost 3001
Reading Console

- ttywatch
  - When snowball is power cycled
    - Resets USB0
    - breaks connection with ttywatch

- Direct read from serial

  `stty -F /dev/ttyUSB0 115200 parodd; cat /dev/ttyUSB0`
Reading Console

- Can't just use "cat"
  - ktest.pl will also get confused on power reset.

- mkfifo snowball-cat

- Make a script "console" that does

```bash
while ;; do
  stty -F /dev/ttyUSB0 115200 parodd 2>/dev/null &&
  cat /dev/ttyUSB0
done > snowball-cat

./console &

CONSOLE = cat ${THIS_DIR}/snowball-cat
Start

- Run ktest.pl with no option, or minimum configs
  - Asks the minimum of questions
  - creates a file ktest.conf
  - defaults to randconfig build
- Update the config to suite your needs
  - use sample.conf
  - Wait for more documentation to come
    - On my high priority TODO list
      - (look for an article on LWN.net)
Options

- **TEST_TYPE** = <what to do>
  - build, install, or boot?
- **MACHINE** = <name-of-board>
  - Unique identifier for board
  - Used for commands (like scp files to target)
- **BUILD_DIR** = <path>
  - directory of the source code (or git repo)
- **OUTPUT_DIR** = <path>
  - directory to build the code “make O=<path>”
Options

- **BUILD_OPTIONS** = <options>
  - Added to make of vmlinux
  - Add -j8 to speed up the build
  - Add targets when needed “bzImage”

- **POWER_CYCLE** = <shell-command>
  - Used to power cycle board
    - for kernel crash
    - failed to “ssh user@$\{MACHINE\} reboot”
Options

- **CONSOLE = <shell-command>**
  - Reads anything that produces stdout of the target's console
  - Must be continuous stream (not reset on reboot)
- **SSH_USER = <user>** (usually “root”)
  - Privileged user on target that can reboot and install kernel images
- **BUILD_TARGET = <relative path to image>**
  - Path relative to OUTPUT_DIR
  - arch/x86/boot/bzImage
Options

- **TARGET_IMAGE** = `<path-to-boot-from>`
  - `/boot/vmlinux-test`
- **LOCAL_VERSION** = `<text>`
  - `localversion file`
  - required to prevent you from killing the stable kernel
Options

- **REBOOT_TYPE = grub** (default)
  - 'script' lets you define how to reboot to kernel

- **REBOOT_SCRIPT = <script>**
  - script to use when REBOOT_TYPE = script

- **GRUB_MENU = <menu title>**
  - searches for this title in /boot/grub/menu.lst
    - grub2 is not yet supported
      - I don't use it ;-) (patches welcomed)
Setup for Snowball

- TEST_TYPE = boot
- MACHINE = snowball (what you ssh to)
- BUILD_DIR = ${THIS_DIR}/linux.git
  - THIS_DIR is a special variable that is defined as the location you are running this
- OUTPUT_DIR = ${THIS_DIR}/snowball-build
- BUILD_OPTIONS = -j8 uImage
- POWER_CYCLE =

  wget --no-proxy -O /dev/null -q --auth-no-challenge 'http://admin:admin@power/outlet?1=CCL'
Setup for Snowball

- TEST_TYPE = boot
- MACHINE = snowball (what you ssh to)
- BUILD_DIR = ${THIS_DIR}/linux.git
  - THIS_DIR is a special variable that is defined as the location you are running this
- OUTPUT_DIR = ${THIS_DIR}/snowball-build
- BUILD_OPTIONS = -j8 uImage
- POWER_CYCLE = echo use the thumb luke; read a
Setup for Snowball

- CONSOLE = cat ${THIS_DIR}/snowball-cat
- SSH_USER = root (but we are not using it)
- BUILD_TARGET = arch/arm/boot/uImage
- TARGET_IMAGE = /var/lib/tftboot/snowball-image
- LOCALVERSION = -test
- REBOOT_TYPE = script
Demo
Options

• **LOG_FILE = <file>**
  - writes all console output and commands run to a file

• **CLEAR_LOG = 1**
  - Overwrites log file at start of test
  - ' = 0' appends to log file (default)
Extra Options

• LOG_FILE = ${OUTPUT_DIR}/snowball.log
Extra Options

- LOG_FILE = ${OUTPUT_DIR}/snowball.log

DEMO
Options

- **MAKE_CMD = <command>** (default “make”)
  - Used to run all makes in ktest.pl
  - make ARCH=powerpc

- **BUILD_TYPE = <type>**
  - pass to make, like “randconfig”
    - BUILD_TYPE = randconfig
      - make randconfig
    - BUILD_TYPE = oldconfig
    - BUILD_TYPE = allnoconfig
  - useconfig:<path/to/config>
    - BUILD_TYPE = useconfig:{$PWD}/myconfig
Extra Options

- Install mkimage (yum install uboot-tools)

- **MAKE_CMD** =

  ```
  PATH=/usr/local/gcc-4.5.2-nolibc/arm-unknown-linux-gnueabi/bin:$PATH
  CROSS_COMPILE=arm-unknown-linux-gnueabi-
  make ARCH=arm
  ```

- **BUILD_TYPE** = u8500_defconfig
  
  - Option used to create config file
  
  - oldconfig
  
  - useconfig:<path-to-config>
DEMO
Config file

- Broken up into sections
  - DEFAULTS
    - All options here are used by all tests
    - Multiple sections are the same as a single section
      - except when a section is conditional
  - TEST_START
    - May override DEFAULTS options
    - Each section defines a single test
      - may have an iterator.
  - Options before first section header
    - defaults to DEFAULTS
Options and Variables

- **OPTION = value**
  - only one definition of an option is allowed in a section
  - used by ktest.pl as commands
  - when defined in TEST_START, only for that test

- **VARIABLE := value**
  - can be overridden throughout the file
  - Used only for reading config file
  - not used by ktest.pl
  - defined in tests are still available in DEFAULTS
Options and Variables

- Defined with '=' or ':=' for option or variable respectively
- both can be used with ${VAR}
  - MACHINE = mybox
  - SSH := ssh root@$MACHINE
  - TEST = ${SSH} /work/test
SKIP

• Sections marked with SKIP are ignored
  - DEFAULTS SKIP
  - TEST_START SKIP

• It is treated like the section has been commented out

• Even variables within a skipped section is not processed (they too are ignored).
ITERATE

- Run the same test over and over
  - TEST_START ITERATE 10
    - just like cut and pasting the TEST_START section 10 times in a row
- TEST_START ITERATE 10 SKIP
  - Just like normal sections, will be skipped and ignored
OVERRIDE

• Allows a section to set options that have been previously set
  - Only works with DEFAULTS section
  - DEFAULTS OVERRIDE

• Rule still applies
  - option may only be defined once within the section

• Overrides options from previous sections
  - later sections can not duplicate options
Check on Demo
Snowball

- SCP_TO_TARGET =
  
  `scp $SRC_FILE $SSH_USER@$MACHINE:$DST_FILE`
  
  - Used to copy files from host to target
Snowball

- SCP_TO_TARGET =
  
  ```
  scp $SRC_FILE $SSH_USER@$MACHINE:$DST_FILE
  ```
  - Used to copy files from host to target

- SCP_TO_TARGET = echo “don't do scp”
Snowball

- **BUILD_NOCLEAN = 1**
  - Does not perform a “make mrproper”

- **CLEAR_LOG = 1**
  - “= 0” appends to LOG_FILE (default)
  - “= 1” truncates file (open with “O_TRUNC”)
  - DEFAULTS option (ignored in TEST_START)
DEMO
Snowball

• ktest.pl will try to install modules if
  - CONFIG_MODULES=y
  - Requires ssh access to target

• No ssh access

• No modules needed
Options

- **MIN_CONFIG = <file>**
  - Best if it is the minimum config to build kernel

- **ADD_CONFIG = <file1> <file2> <file3>**
  - Add configs to MIN_CONFIG
    - MIN_CONFIG takes precedence

- Both set and unset configs take affect
  - common mistake is to keep the
    - # CONFIG_FOO_BAR is not set
      - grep '^CONFIG' .config > min_config
Snowball

- ADD_CONFIG = ${THIS_DIR}/addconfig
  - # CONFIG_MODULES is not set
Snowball

- ADD_CONFIG = ${THIS_DIR}/addconfig
  - # CONFIG_MODULES is not set

DEMO
IF

- Sections may be conditionally skipped
  - TEST_START_IF_${VAR}
    - will only run if VAR is defined and is non zero

- May also handle compares
  - TEST_START_IF_${TEST_CNT} > 10

- Complex compares
  - TEST_START_IF_${DEFAULTS} ||
    (${TEST_RUN} == ARM)
  - (Note: does not handle line breaks)
IF

• DEFINED
  - Test if a variable or option is defined
    • DEFAULTS IF DEFINED REBOOT

• NOT DEFINED
  - test if a variable is not defined
    • DEFAULTS IF NOT DEFINED BITS
      - BITS := 64
    • TEST = ./hackbench_${BITS} 10
ELSE (IF)

- Followed by a section that has an IF
  - DEFAULTS IF \( \text{ARCH} \) == x86_64
    - BITS := 64
  - DEFAULTS ELSE
    - BITS := 32
- May be followed by IF to do selections
  - DEFAULTS IF \( \text{TEST} \) == build
  - DEFAULTS ELSE IF \( \text{TEST} \) == boot
  - DEFAULTS ELSE
INCLUDE

- INCLUDE <file>
  - can be full path
  - searches config file directory
  - searches local director

- Only allowed in DEFAULTS section
- may define TEST_START
- DEFAULTS defined before are seen
- DEFAULTS defined in included files are defined in parent file (like CPP)
MACHINE = mxtest
BOX := mxtest
CONSOLE = nc -d fedora 3001

# TESTS = patchcheck, randconfig, boot, test, config-bisect, biscet
TEST := patchcheck
MULTI := 0

# Run allno, ftrace,noftrace, notrace, allmod and allyes
CONFIG_TESTS := 1
CONFIG_ALLYES := 0
CONFIG_ALLYES_TEST_TYPE := build

CHECKOUT :=
# REBOOT = none, fail, empty
#REBOOT := fail

MACHINE := mxtest

GCC_VERSION := 4.6.0

BITS := 64

INCLUDE include/defaults.conf
INCLUDE include/patchcheck.conf
INCLUDE include/tests.conf
INCLUDE include/bisect.conf
INCLUDE include/config-bisect.conf
INCLUDE include/minconfig.conf
INCLUDE include/config-tests.conf

DEFAULTS OVERRIDE
POST_INSTALL =
OUTPUT_DIR = ${THIS_DIR}/nobackup/${MACHINE}
```
# defaults.conf

DEFAULTS IF NOT DEFINED BITS
BITS := 64

DEFAULTS

SSH := ssh ${SSH_USER}@${MACHINE}
THIS_DIR := /home/rostedt/work/git
CONFIG_DIR := ${THIS_DIR}/configs/${MACHINE}

REBOOT_SUCCESS_LINE = login:

BUILD_DIR = ${THIS_DIR}/linux-${BOOT_TYPE}.git
OUTPUT_DIR = ${THIS_DIR}/nobackup/${MACHINE}/${BOOT_TYPE}

DEFAULTS
REBOOT_ON_SUCCESS = 0
REBOOT_ON_ERROR = 1
POWEROFF_ON_ERROR = 0
POWEROFF_ON_SUCCESS = 0

DEFAULTS
SSH_USER = root
POWER_OFF = ${THIS_DIR}/${MACHINE}-poweroff
POWER_CYCLE = ${THIS_DIR}/${MACHINE}-cycle
BUILD_TARGET = arch/x86/boot/bzImage
CLEAR_LOG = 1
LOCALVERSION = -test
MAKE_CMD = GCC_VERSION=${GCC_VERSION} distmake-${BITS}
BUILD_OPTIONS = -j40
LOG_FILE = ${THIS_DIR}/nobackup/${MACHINE}/${MACHINE}.log
MIN_CONFIG = ${CONFIG_DIR}/config-min
TMP_DIR = /tmp/ktest/${MACHINE}

GRUB_MENU = ${GRUBNAME} Kernel
TARGET_IMAGE = /boot/vmlinuz-test${EXT}
POST_INSTALL = ${SSH} /sbin/dracut -f /boot/initramfs-test${EXT}.img $KERNEL_VERSION

STORE_FAILURES = ${THIS_DIR}/failures/${MACHINE}
```
TEST_START

• build
  - just builds the kernel

• boot
  - builds and boots the kernel

• test
  - builds, boots and runs a command
  - TEST = <command>
    • runs from host but may use 'ssh' to target
TEST_START IF ${TEST} == boot
TEST_TYPE = boot
BUILD_TYPE = oldconfig
BUILD_NOCLEAN = 1

TEST_START ITERATE 10 IF ${TEST} == randconfig
MIN_CONFIG = ${CONFIG_DIR}/config-net
TEST_TYPE = test
BUILD_TYPE = randconfig
TEST = ${SSH} /work/c/hackbench_${BITS} 50

TEST_START ITERATE 10 IF ${TEST} == randconfig && ${MULTI}
TEST_TYPE = boot
BUILD_TYPE = randconfig
MIN_CONFIG = ${CONFIG_DIR}/config-min
MAKE_CMD = make

TEST_START IF ${TEST} == test
TEST_TYPE = test
#BUILD_TYPE = oldconfig
#BUILD_TYPE = useconfig:${CONFIG_DIR}/config-net
BUILD_TYPE = useconfig:${CONFIG_DIR}/config-bisect
#BUILD_TYPE = nobuild
TEST = ${SSH} /work/bin/test-mod-event
BUILD_NOCLEAN = 1
TEST_START

- patchcheck
  - Requires BUILD_DIR be a git repo
  - PATCHCHECK_TYPE = <type>
    - build, boot or test
  - PATCHCHECK_START = <commit>
    - git commit to start testing (SHA1, tag, etc)
  - PATCHCHECK_STOP = <commit>
    - git commit to stop (SHA1, HEAD)
PATCH_START := HEAD~1
PATCH_END := HEAD
CHECKOUT := trace(trace/tip/perf/core
PATCH_CONFIG = `${CONFIG_DIR}/config-ftrace-patchcheck
PATCH_TEST := `${SSH} "/work/bin/trace-cmd-filter-stress && trace-cmd record -e all -p function -l schedule /work/c/hackbench_${BITS} 50 && trace-cmd report && /work/bin/test-mod-event"

TEST_START IF ${TEST} == patchcheck
TEST_TYPE = patchcheck
MIN_CONFIG = `${PATCH_CONFIG}

TEST = `${PATCH_TEST}
PATCHCHECK_TYPE = test
PATCHCHECK_START = `${PATCH_START}
PATCHCHECK_END = `${PATCH_END}
CHECKOUT = `${CHECKOUT}

TEST_START IF ${TEST} == patchcheck && ${MULTI}
TEST_TYPE = patchcheck
MIN_CONFIG = `${PATCH_CONFIG}
TEST = `${PATCH_TEST}
PATCHCHECK_TYPE = test
PATCHCHECK_START = `${PATCH_START}
PATCHCHECK_END = `${PATCH_END}
CHECKOUT = `${CHECKOUT}
MAKE_CMD = GCC_VERSION=4.5.1 distmake-64

TEST_START IF ${TEST} == patchcheck && ${MULTI}
TEST_TYPE = patchcheck
MIN_CONFIG = `${PATCH_CONFIG}
TEST = `${PATCH_TEST}
PATCHCHECK_TYPE = test
PATCHCHECK_START = `${PATCH_START}
PATCHCHECK_END = `${PATCH_END}
CHECKOUT = `${CHECKOUT}
MAKE_CMD = make
TEST_START

- bisect
  - Requires BUILD_DIR to be a git repo
  - performs a git bisect
  - BISECT_TYPE (build, boot or test)
  - BISECT_GOOD = <commit>
    - git commit that is marked good
      - (git bisect good <commit>)
  - BISECT_BAD = <commit>
    - git commit that is marked bad
      - (git bisect bad <commit>)
TEST_START

- bisect
  - BISECT_REVERSE = 1
    - good is bad, bad is good
  - BISECT_MANUAL = 1
    - asks you between tests if bisect was good
  - BISECT_CHECK = 1 (good/bad)
    - tests good and bad before doing bisect
  - BISECT_FILES = <file1> <file2> <dir1>
    - Only bisect based on these files or directories
    - runs 'git bisect start -- <file1> <file2> <dir1>'
TEST_START

- bisect
  - BISECT_SKIP = 0
    - fail on failed bisect instead of running
      - git bisect skip
  - BISECT_REPLY = <file>
    - failed bisect, run git bisect log > file
  - BISECT_START = <commit>
    - checks out commit after bisect start and stop
    - runs after BISECT_REPLY if it is defined
  - MIN_CONFIG = <config>
    - future will allow BUILD_TYPE
TEST_START_IF ${TEST} == bisect
TEST_TYPE = bisect
BISECT_TYPE = boot
MIN_CONFIG = ${CONFIG_DIR}/config-ftrace-patchcheck
BISECT_GOOD = v2.6.39
BISECT_BAD = HEAD
CHECKOUT = origin/master
TEST = ssh ${USER}@${MACHINE} /work/bin/test-writeback-sync
#BISECT_REPLAY = /tmp/replay1
Check on DEMO
Options

• POST_INSTALL = <what to do after install>
  - optional
    - ssh user@target /sbin/dracut -f /boot/initramfs-test.img $KERNEL_VERSION
  - $KERNEL_VERSION is not a normal variable
    • does not have {} 
    • it is replaced by the kernel version found by ktest.pl
Options

• SWITCH_TO_TEST = <shell-command>
  - Run before rebooting to test kernel
• SWITCH_TO_GOOD = <shell-command>
  - Run before rebooting to default kernel
Snowball

- TFTPBOOT := /var/lib/tftpboot
- TFTPDEF := ${TFTPBOOT}/snowball-default
- TFTPTEST := ${OUTPUT_DIR}/${BUILD_TARGET}
- SWITCH_TO_TEST = cp ${TFTPTEST} ${TARGET_IMAGE}
- SWITCH_TO_GOOD = cp ${TFTPDEF} ${TARGET_IMAGE}
DEMO
Options

- **SUCCESS_LINE** = <text-denoting-success>
  - default “login:”
  - Can change to “root@linaro:~#”

- **REBOOT_SUCCESS_LINE** = <text>
  - Quick way to detect successful good reboot
Options

- `POWEROFF_ON_SUCCESS = 1`
- `REBOOT_ON_SUCCESS = 1`
  - ignored if `POWER_OFF_ON_SUCCESS` is set
- `POWEROFF_ON_ERROR = 1`
- `REBOOT_ON_ERROR = 1`
  - ignored if `POWEROFF_ON_ERROR` is set
- `POWERCYCLE_AFTER_REBOOT = <secs>`
  - nice when reboot doesn't finish the reboot
- `POWEROFF_AFTER_HALT = <secs>`
Options

- **DIE_ON_FAILURE = 0 (default 1)**
  - When set to zero, a failed test will not stop ktest

KTEST RESULT: TEST 1 SUCCESS!!!!  **

KTEST RESULT: TEST 2 Failed: failed - got a bug report
Options

- STORE.FAILURES = <dir>
  - Used when DIE_ON_FAILURE = 0
  - Creates directory within this directory
    - MACHINE-TEST-fail-TIMESTAMP
    - mitest-boot-randconfig-fail-20110008154933
  - Saves dmesg, config, and build log
• config_bisect
  - Find a bad config in a config file
  - CONFIG_BISECT_TYPE (build, boot, test)
  - CONFIG_BISECT_GOOD = <file> (optional)
    • start config
    • default is to use MIN_CONFIG
    • The current good is saved in the OUTPUT_DIR as “config_good”
  - CONFIG_BISECT = <file>
    • the bad config
    • must be superset of good config file
config_bisect

- How it works?
  - ignore configs defined in good config
  - try first half
    - test if it changed config
    - test other half
      - only one config needs to be set to continue
  - test passes
    - Permanently enable configs that are set
  - test fails
    - have new bad config
      - repeat
TEST_START_IF ${TEST} == config-bisect
TEST_TYPE = config_bisect
CONFIG_BISECT_TYPE = boot
#CONFIG_BISECT = ${THIS_DIR}/nobackup/failures/mxtest-boot-randconfig-fail-20110502120128/config
CONFIG_BISECT = ${THIS_DIR}/config-bad
#CHECKOUT = origin/master
#CONFIG_BISECT_GOOD = ${THIS_DIR}/config-good
• **make_min_config**
  - OUTPUT_MIN_CONFIG = <file>
    - The new min config
  - START_MIN_CONFIG = <file> (optional)
    - default uses MIN_CONFIG
  - IGNORE_CONFIG = <file> (optional)
    - Only configs that ktest.pl found fails to boot
    - Does not add allnoconfig configs
    - Does not add selected configs
make_min_config

• How it works?
  - Read Kconfigs to find depends and selects
  - Pick the config which has the most depending on it
  - Disable that config (make sure new config changes)
  - Passes – disable it and all that depend on it
    • Update OUTPUT_MIN_CONFIG
  - Fails – Keep it permanently enabled
    • Add to IGNORE_CONFIG
cross compiling

- Get binary cross compilers from kernel.org
  - http://www.kernel.org/pub/tools/crosstool/files/bin/x86_64/
  - http://artfiles.org/kernel.org/pub/tools/crosstool/files/bin/x86_64/4.5.2/

- All developers should run cross compilers for all the archs their code affects (even drivers)
# crosstests.conf

THIS_DIR := /work/autotest
ARCH_DIR := ${THIS_DIR}/nobackup/linux-test.git/arch

BUILD_DIR = ${THIS_DIR}/nobackup/cross-linux.git

DO_FAILED := 0
DO_DEFAULT := 1
#RUN := m32r

GCC_VER = 4.5.2
MAKE_CMD = PATH=/usr/local/gcc-${GCC_VER}-nolibc/${CROSS}/bin:$PATH CROSS_COMPILE=${CROSS}- make ARCH=${ARCH}
TEST_TYPE = build

BUILD_TYPE = defconfig

TEST_NAME = ${ARCH} ${CROSS}

# alpha
TEST_START IF ${RUN} == alpha || ${DO_DEFAULT}
#MIN_CONFIG = ${ARCH_DIR}/alpha/defconfig
CROSS = alpha-linux
ARCH = alpha

# arm
TEST_START IF ${RUN} == arm || ${DO_DEFAULT}
#MIN_CONFIG = ${ARCH_DIR}/arm/configs/cm_x300_defconfig
CROSS = arm-unknown-linux-gnueabi
ARCH = arm

# black fin
TEST_START IF ${RUN} == bfin || ${DO_DEFAULT}
#MIN_CONFIG = ${ARCH_DIR}/blackfin/configs/BF561-EZKIT-SMP_defconfig
CROSS = bfin-uclinux
ARCH = blackfin
BUILD_OPTIONS = -j8 vmlinux
# crosstests.conf

# cris - FAILS?
TEST_START_IF $(RUN) == cris || $(RUN) == cris64 || $(DO_FAILED)
#MIN_CONFIG = {ARCH_DIR}/cris/configs/etraxfs_defconfig
CROSS = cris-linux
ARCH = cris

# cris32 - not right arch?
TEST_START_IF $(RUN) == cris || $(RUN) == cris32 || $(DO_FAILED)
#MIN_CONFIG = {ARCH_DIR}/cris/configs/etrax-100lx_v2_defconfig
CROSS = crisv32-linux
ARCH = cris

# ia64
TEST_START_IF $(RUN) == ia64 || $(DO_DEFAULT)
#MIN_CONFIG = {ARCH_DIR}/ia64/configs/generic_defconfig
CROSS = ia64-linux
ARCH = ia64

# frv
TEST_START_IF $(RUN) == frv || $(DO_FAILED)
CROSS = frv-linux
ARCH = frv
GCC_VER = 4.5.1

# h8300 - failed make defconfig??
TEST_START_IF $(RUN) == h8300 || 0
CROSS = h8300-elf
ARCH = h8300
GCC_VER = 4.5.1

# m68k fails with error?
TEST_START_IF $(RUN) == m68k || $(DO_DEFAULT)
#MIN_CONFIG = {ARCH_DIR}/m68k/configs/multi_defconfig
CROSS = m68k-linux
ARCH = m68k
crosstests.conf

[ ... ]

TEST_START IF ${RUN} == x86 || ${RUN} == i386 || ${DO_DEFAULT}
MAKE_CMD = distmake-32
ARCH = i386
CROSS =

TEST_START IF ${RUN} == x86 || ${RUN} == x86_64 || ${DO_DEFAULT}
MAKE_CMD = distmake-64
ARCH = x86_64
CROSS =

DEFAULTS
MACHINE = crosstest
SSH_USER = root
OUTPUT_DIR = ${THIS_DIR}/nobackup/cross-compile
BUILD_TARGET = cross
TARGET_IMAGE = image
POWER_CYCLE = cycle
CONSOLE = console
LOCALVERSION = version
GRUB_MENU = grub
LOG_FILE = ${THIS_DIR}/nobackup/cross-compile/cross.log
BUILD_OPTIONS = -j8

REBOOT_ON_ERROR = 0
POWEROFF_ON_ERROR = 0
POWEROFF_ON_SUCCESS = 0
REBOOT_ON_SUCCESS = 0
DIE_ON_FAILURE = 0
STORE_FAILURES = ${THIS_DIR}/nobackup/failures/cross

CLEAR_LOG = 1
TODO

- Make initial (no arg) have better comments
  - Done! (v3.3)

- bisect and config bisect can restart without user manually saving it
  - Done! (v3.3)

- If all tests is just build, do not require options for boot and test
  - Done! (v3.3)
TODO

- Add output results to all tests
- Fix bisects to use BUILD_TYPE
- Add option to change SIGINT to CONSOLE
- Change config-bisect to diff any two configs