Kernel Development

> WIP work in progress

An example of kernel development with `flake.nix` .

```nix
{
  description = "NixOS running on LicheePi 4A";

  inputs = {
    nixpkgs.url = "github:nixos/nixpkgs/nixos-24.11-small";

    # custom kernel's source
    thead-kernel = {
      url = "github:revyos/thead-kernel/lpi4a";
      flake = false;
    };
  };

  outputs = inputs@{
    self
    ,nixpkgs
    ,thead-kernel
    ,... }:
  let
    pkgsKernel = import nixpkgs {
      localSystem = "x86_64-linux";
      crossSystem = {
        config = "riscv64-unknown-linux-gnu";
      };

      overlays = [
        (self: super: {
          # use gcc 13 to compile this custom kernel
          linuxPackages_thead = super.linuxPackagesFor (super.callPackage ./pkgs
            src = thead-kernel;
            stdenv = super.gcc13Stdenv;
            kernelPatches = with super.kernelPatches; [
              bridge_stp_helper
              request_key_helper
            ];

          });
        })
      ];
    };
  in
  {
    nixosConfigurations.lp4a = nixpkgs.lib.nixosSystem {
      system = "x86_64-linux";

      specialArgs = {
        inherit nixpkgs pkgsKernel;
      };
      modules = [
        {
          # cross-compile this flake.
          nixpkgs.crossSystem = {
            system = "riscv64-linux";
          };
        }

        ./modules/licheepi4a.nix
        ./modules/sd-image-lp4a.nix
      ];
    };

    # use `nix develop .#kernel` to enter the environment with the custom kernel
    # and then use `unpackPhase` to unpack the kernel source code and cd into it
    # then you can use `make menuconfig` to configure the kernel.
    #
    # problem
    #   - using `make menuconfig` - Unable to find the ncurses package.
    devShells.x86_64-linux.kernel = pkgsKernel.linuxPackages_thead.kernel.dev;

    # use `nix develop .#fhs` to enter the fhs test environment defined here.
    devShells.x86_64-linux.fhs = let
      pkgs = import nixpkgs {
        system = "x86_64-linux";
      };
    in
      # the code here is mainly copied from:
      #   https://wiki.nixos.org/wiki/Linux_kernel#Embedded_Linux_Cross-compile_
      (pkgs.buildFHSUserEnv {
        name = "kernel-build-env";
        targetPkgs = pkgs_: (with pkgs_;
          [
            # we need theses packages to run `make menuconfig` successfully.
            pkgconfig
            ncurses

            pkgsKernel.gcc13Stdenv.cc
            gcc
          ]
          ++ pkgs.linux.nativeBuildInputs);
        runScript = pkgs.writeScript "init.sh" ''
          # set the cross-compilation environment variables.
          export CROSS_COMPILE=riscv64-unknown-linux-gnu-
          export ARCH=riscv
          export PKG_CONFIG_PATH="${pkgs.ncurses.dev}/lib/pkgconfig:"
          exec bash
        '';
      }).env;
  };
}
```

With the above `flake.nix`, I can enter the kernel build environment with `nix develop .#kernel`, and then use `unpackPhase` to unpack the kernel source code and cd into it. But I can't use `make menuconfig` to configure the kernel, because the `ncurses` package is missing in this environment.

To solve this problem, I add a `fhs` environment to install the `ncurses` package and other necessary packages, and then I can use `nix develop .#fhs` to enter this environment and use `make menuconfig` to configure the kernel.

References

- Linux kernel - NixOS Wiki
- https://github.com/jordanisaacs/kernel-module-flake

0 reactions

0 comments