

```
#####  
#####      Raspberry Pi3, 8-16Gb card, Rasbian-Stretch-lite      #####  
#####  
N8AVX - December 2018
```

These instructions will, when followed, produce a working JNOS system ready to configure for use as a personal JNOS station or as a HamGate. The difference is in the autoexec.nos configuration.

This is a plain vanilla install using as much defaults as possible. JNOS is fairly configurable as to where the various bits go, but there is a default place for things. I'll follow the defaults built into the code and installer.

The installer will suggest /jnos as the root of the directory tree for JNOS. It will then place the executables and configuration bits where it wants, which is mostly in the /jnos directory.

I think this is messy. I don't do it that way, but to keep things dead simple, the default way is presented here.

For security, you will be creating a new user for JNOS and setting a new password for the user pi. This is to promote security as pi with the default password is very well known (pi/raspberry).

Let's get to it. I assume you have a Pi3, have put an image of raspbian-stretch-lite on an 8 to 16GB uSD card, and have connected a keyboard and monitor to the Pi.

I also assume that when I say "log in as user pi", you know how to, and what to do after you enter the passwd prompt. In short, you need to be comfortable using Linux.

You should power it up and ensure you can, in fact, log into the Pi.

Once you can do that, we can begin.

I used 2017-11-29-raspbian-stretch-lite.img from the Raspberrypi.org site.

This has been tested with 2018-11-13-raspbian-stretch-lite.img and found to work as well.

System Administration Tasks

Create a user for jnos and add to sudoers

The password for the user pi needs to be changed

Log in as user pi and do the following:

Type the command and follow the prompts:

passwd

The user nosuser needs to be created. If you wish to use a different user, you can

Type the commands:

sudo adduser nosuser

```
Adding user `nosuser' ...
Adding new group `nosuser' (1002) ...
Adding new user `nosuser' (1002) with group `nosuser' ...
Creating home directory `/home/nosuser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: <password>
Retype new UNIX password: <password>
passwd: password updated successfully
Changing the user information for nosuser
Enter the new value, or press ENTER for the default
Full Name []: JNOS
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [Y/n] y
```

Type the command:

sudo adduser nosuser sudo

Type the command:

sudo adduser nosuser adm

We have now created a user for JNOS and put them in the right groups so they can use sudo.

We now need to log out as *pi* and log in as *nosuser*.

Type the command:

exit

Note: Whenever you use the *sudo* command, you may get a request for the users' sudo password. It is the same as the password you use to log into the Pi.

Log in as nosuser

Do the following to prove you can sudo:

Type the following command:

nano /boot/config.txt

You should see near the bottom of the screen that the file is unwriteable

Ctrl-X to exit the editor.

Type the following command:

sudo nano /boot/config.txt

After entering your password (if asked), you will no longer see the "unwriteable" text.

Congratulations, you can sudo. Ctrl-X to exit the editor.

By the way, use Ctrl-O to save any changes you make in a file.

From now on, when you work with the JNOS software and files, use the nosuser user. The "pi" user is still enabled, but since we changed the password, folks can't get in using the default. This is a good thing. In addition, other tutorials on how to do and install things often reference the "pi" user, so this leaves that in there so "Things Still Work".

Update and get software.

This requires an internet connection.

You need to be running the latest software and add some to be able to successfully compile and run JNOS.

Type the following commands:

```
sudo apt-get update  
sudo apt-get upgrade  
sudo reboot
```

Log back in and type the following commands:

```
sudo apt-get install build-essential  
sudo apt-get install libncurses5-dev libssl-dev mc telnet
```

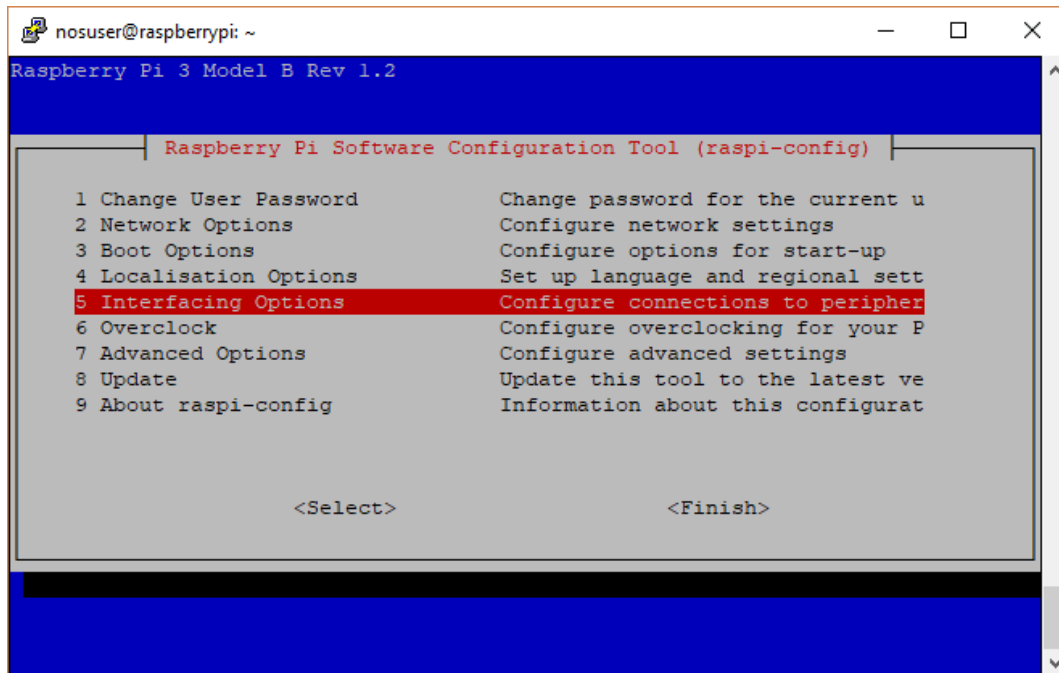
You may get something about a package not available. If it for mc, no problem really, it is just a neat little program for wandering around the directory structure, looking at files and optionally editing them.

Enable SSH, set Timezone, change keyboard

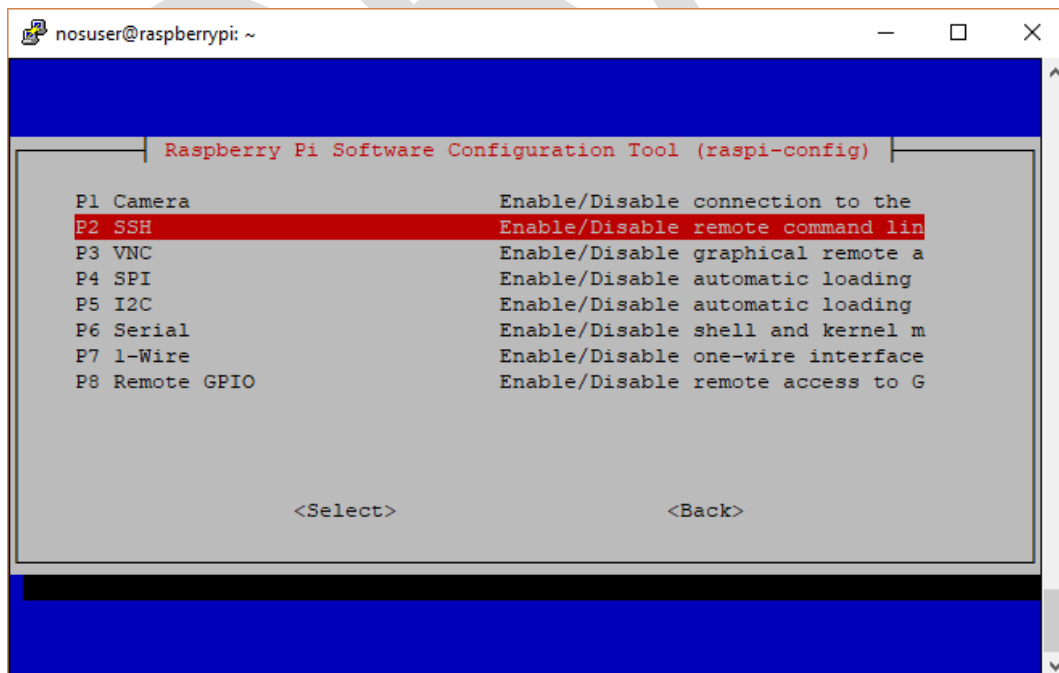
Type the following command:

sudo raspi-config

To enable SSH, use the arrow key to move down to *Interfacing Options*, and then press the Enter key. When using these menus, you may have to scroll up or down with the arrow keys.



Using the arrow key, move down to *SSH* and press the enter key.



JNOS on a Pi v4.3

Using the Tab key, highlight <Yes>, then press the Enter key.

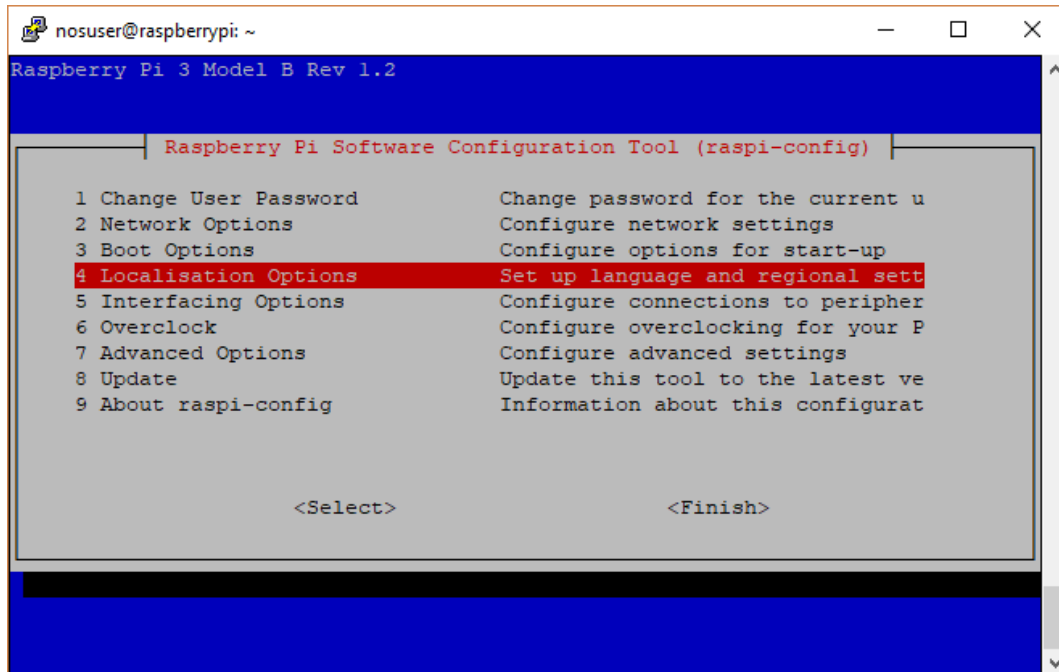


Press the Enter key to get back to the main menu

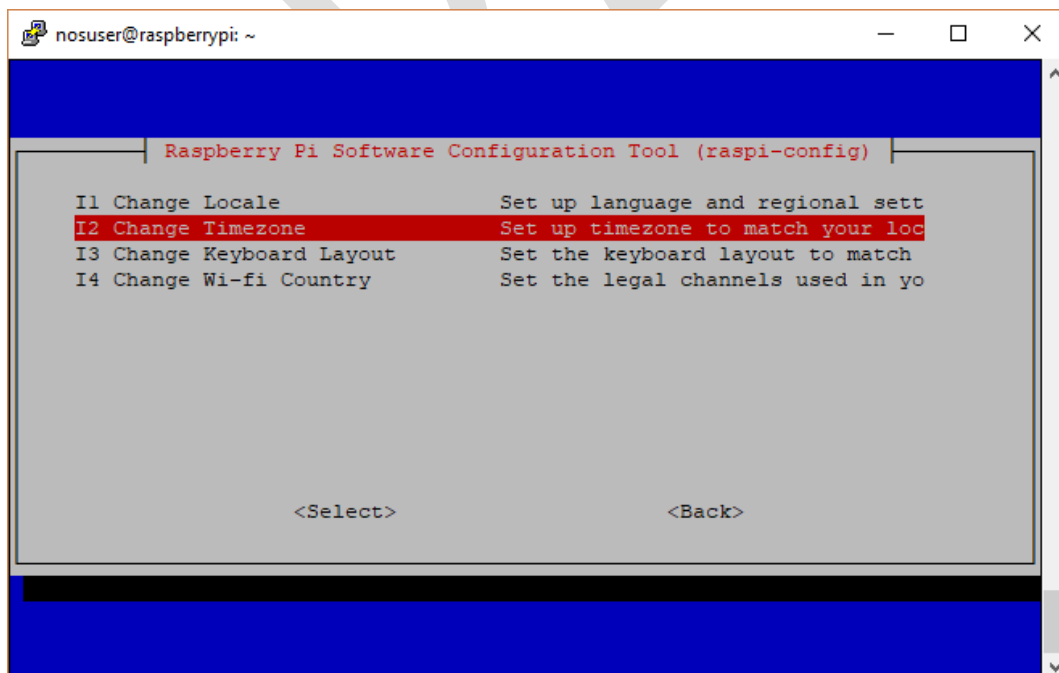


I live in the US Eastern Time zone north of Detroit. The following illustrates **my** choices. Yours may be (and probably are) different. Feel free to choose what is workable for you.

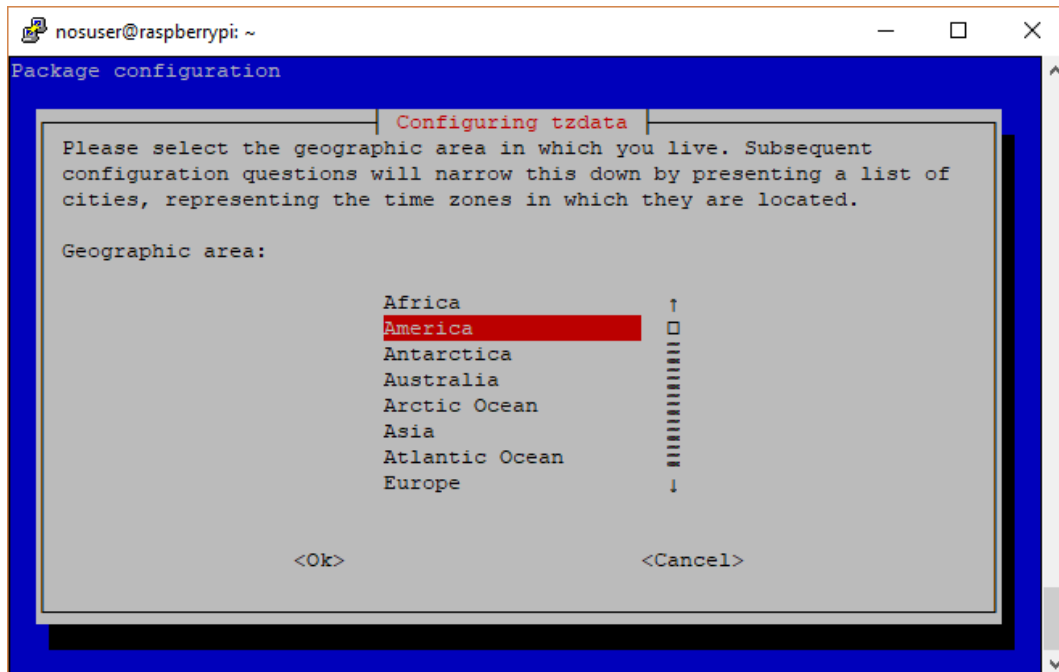
Use the arrow key to move down and highlight *Localization Options*, then press the Enter key.



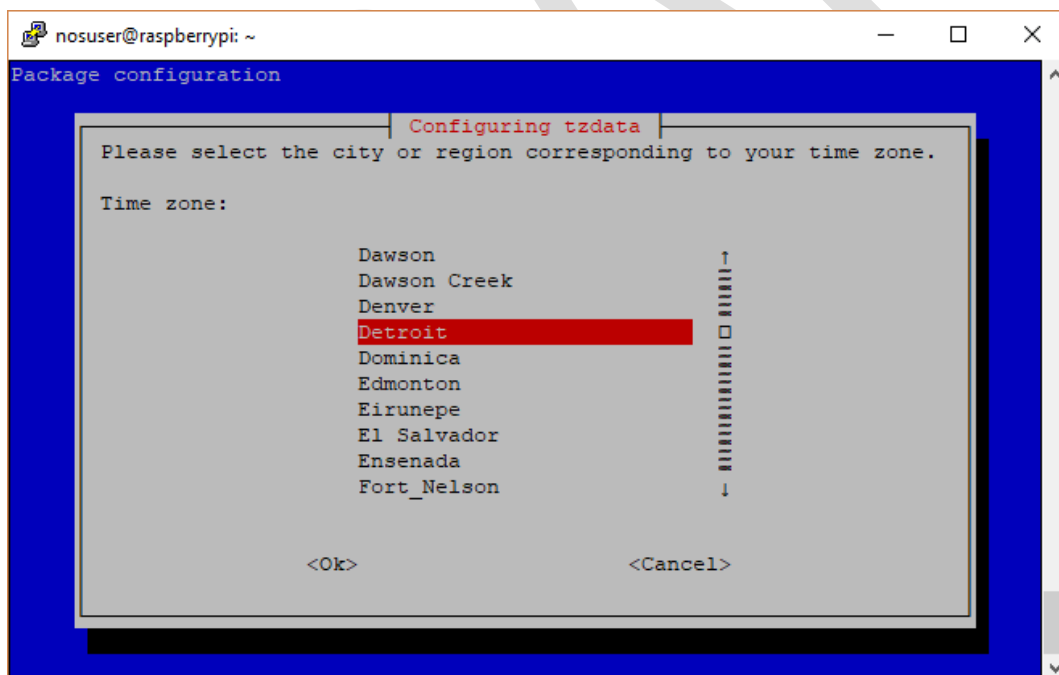
Use the arrow keys to highlight the *Change Timezone* option and press the Enter key.



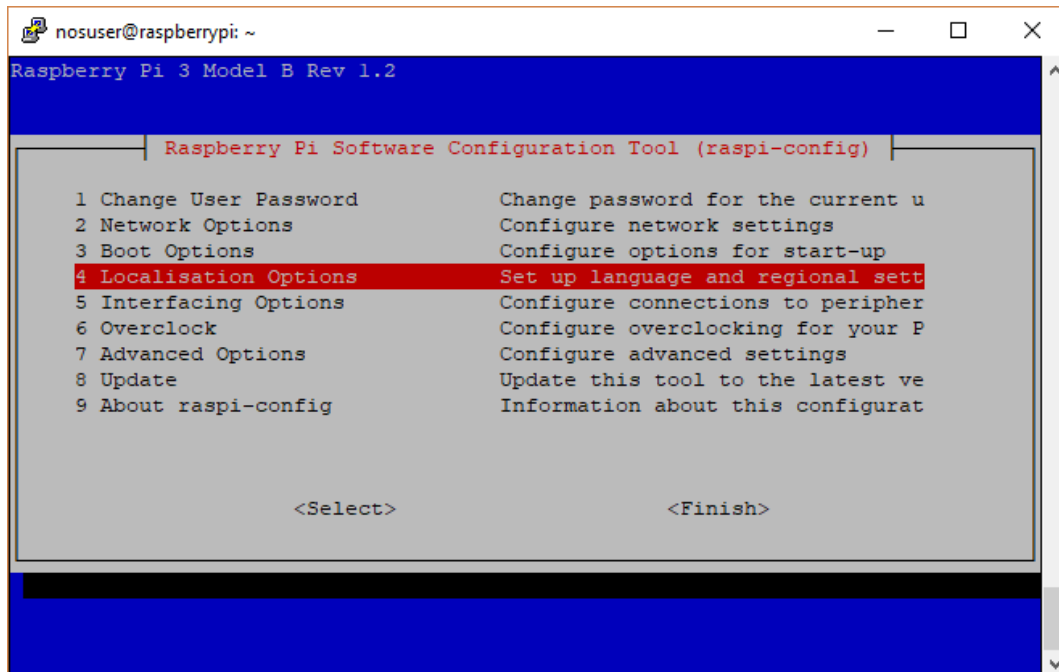
Use the arrow keys to highlight the *America* option and press the Enter key.



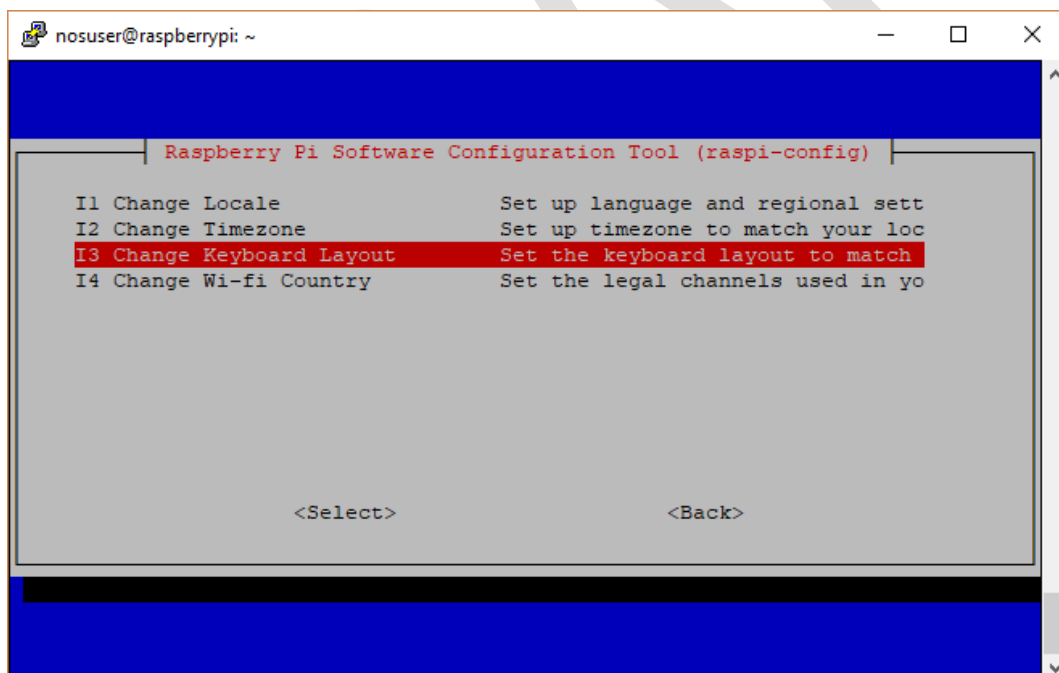
Use the arrow keys to highlight the *Detroit* option and press the Enter key.



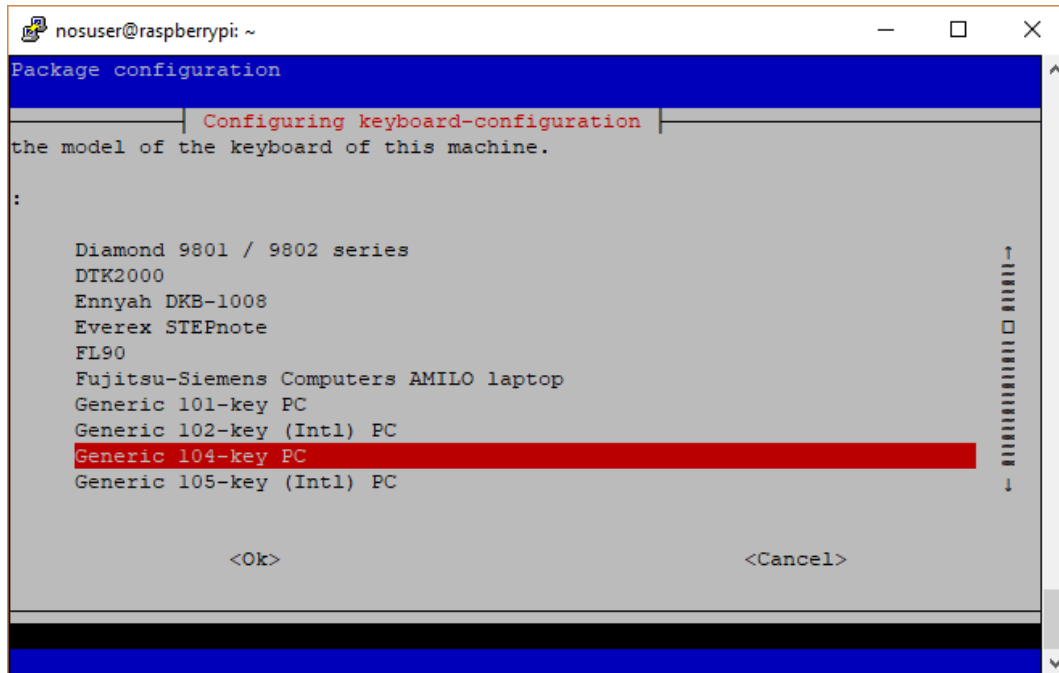
Use the arrow key to move down and highlight *Localization Options*, then press the Enter key.



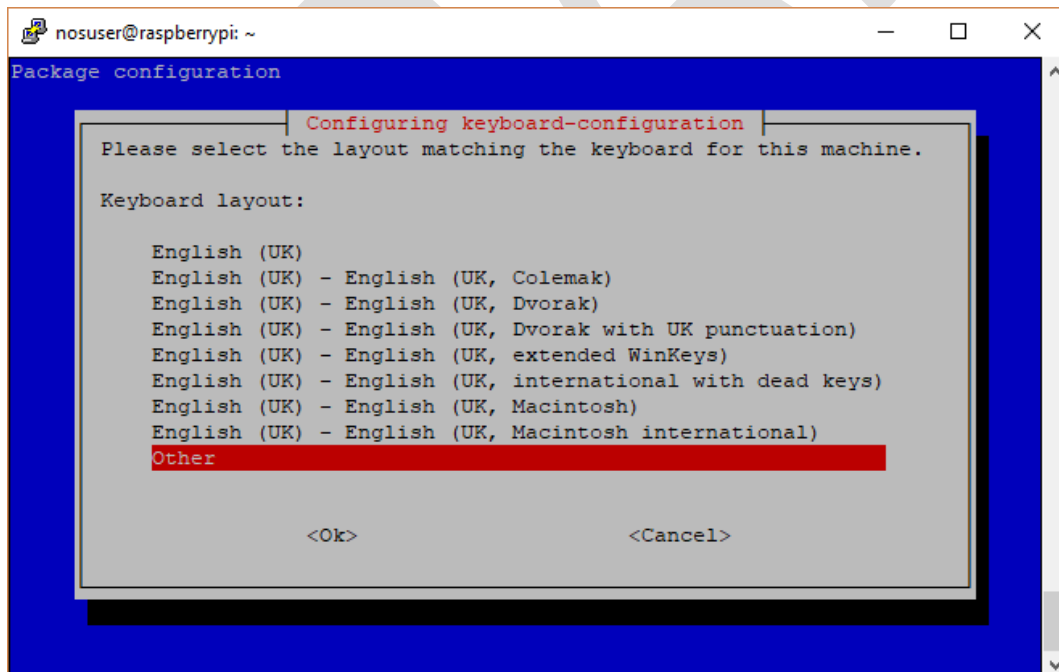
Use the arrow keys to highlight the *Change Keyboard Layout* option and press the Enter key.



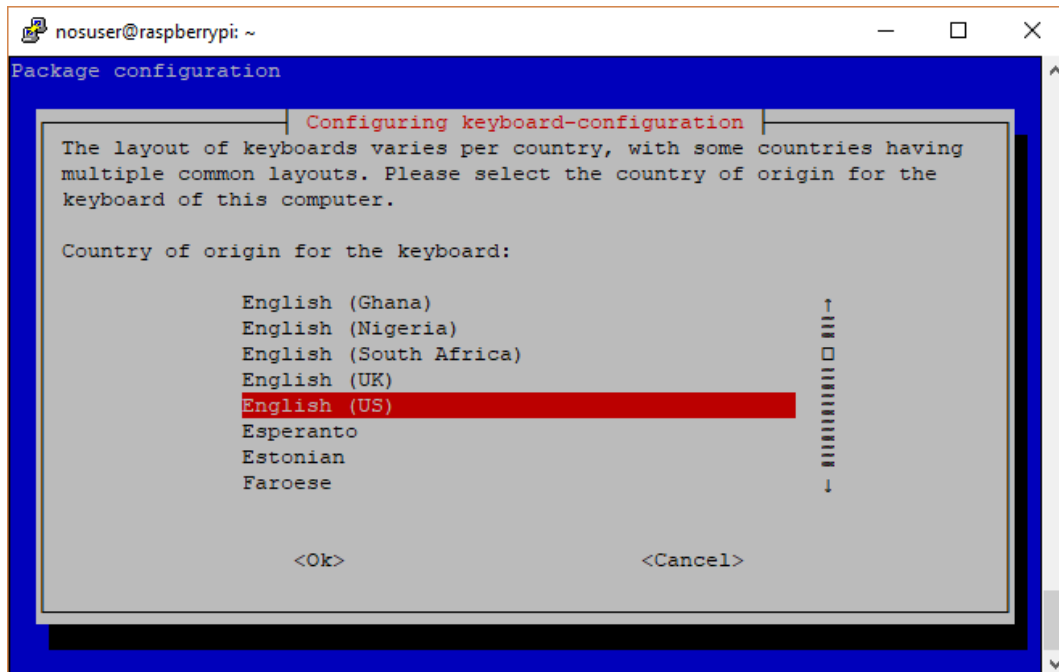
Use the arrow keys to highlight the *Generic 104-key PC* option and press the Enter key.
If you can find your keyboard in the list, feel free to do so. Otherwise, I've found this to be a safe option.



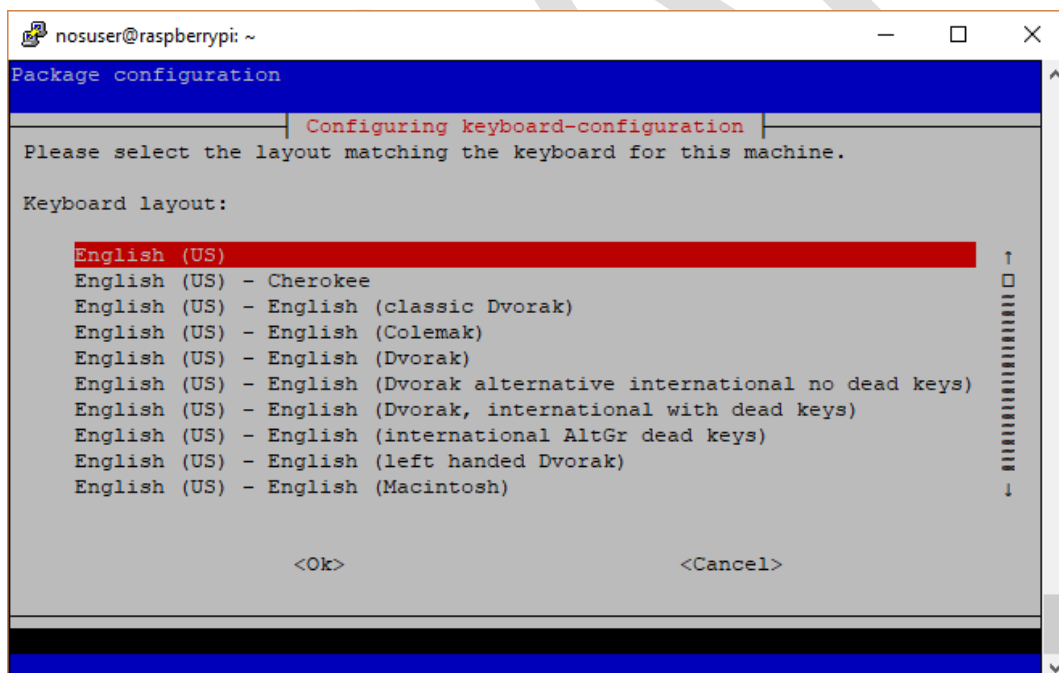
Use the arrow keys to highlight the *Other* option and press the Enter key.



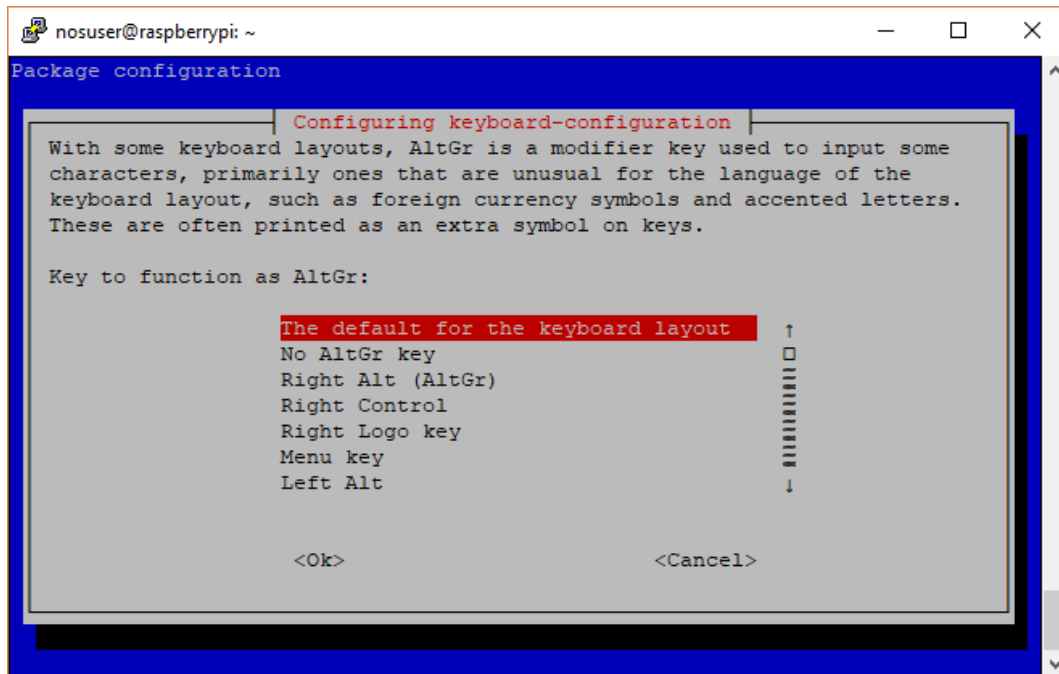
Use the arrow keys to highlight the *English (US)* option and press the Enter key.



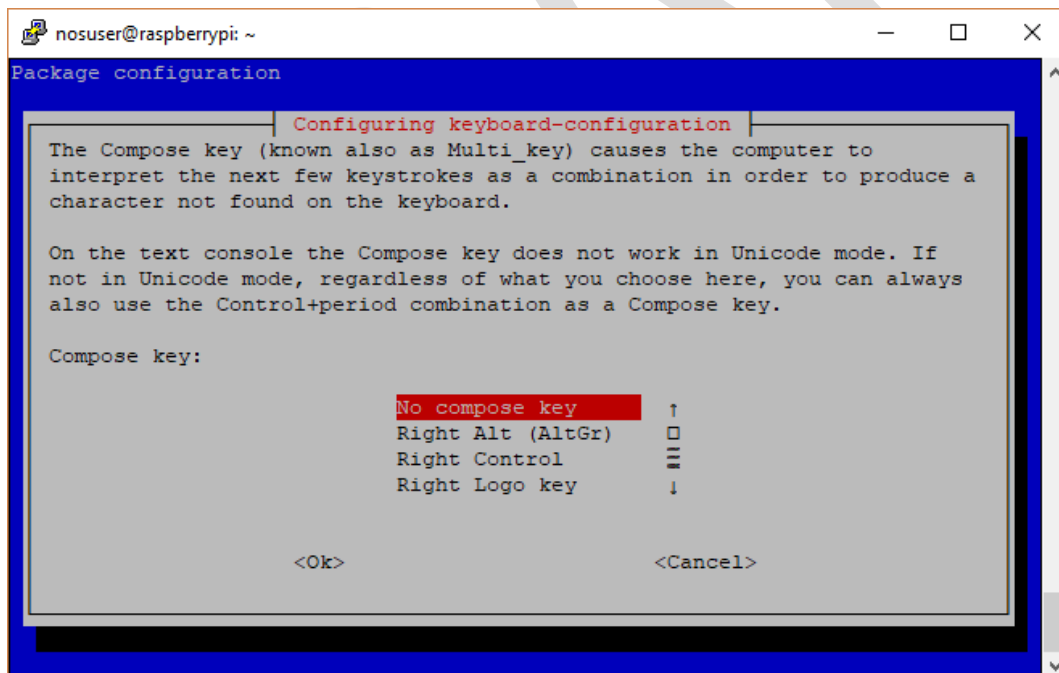
Use the arrow keys to highlight the *English (US)* option and press the Enter key.



Use the arrow keys to highlight *The default for the keyboard layout* option and press the Enter key.

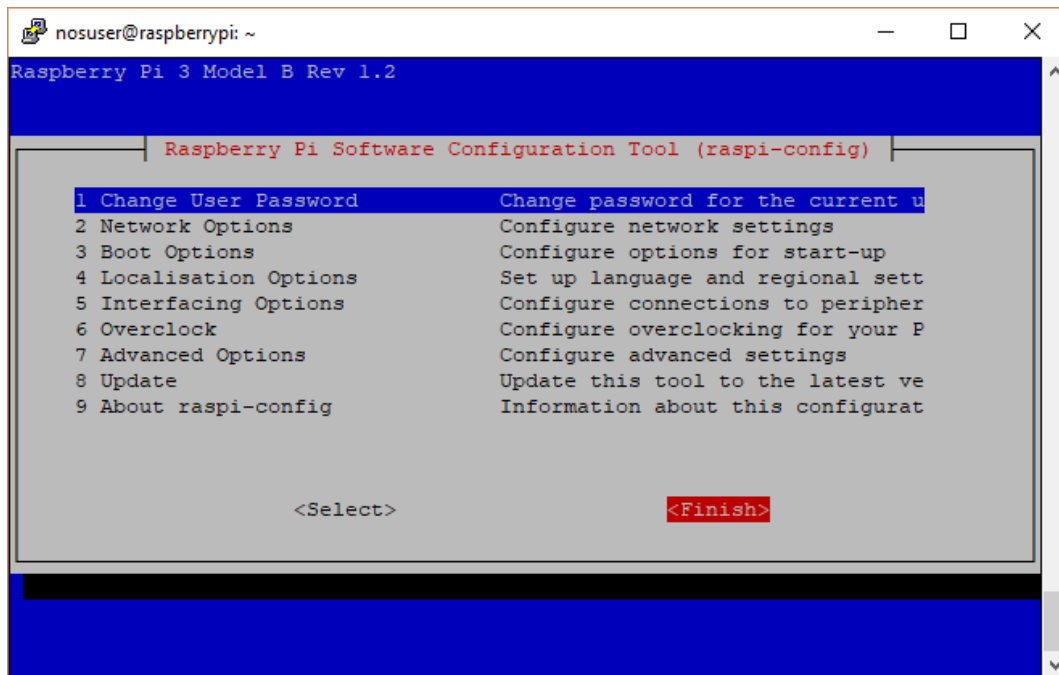


Use the Tab key to highlight the *No Compose Key* option and press the Enter key.



Exit out of raspi-config to save those changes.

Use the Tab key to highlight *<Finish>* and press the Enter key.



Enable IP Forwarding

Now you need to enable IP Forwarding.

Type the following command:

```
sudo nano /etc/sysctl.conf
```

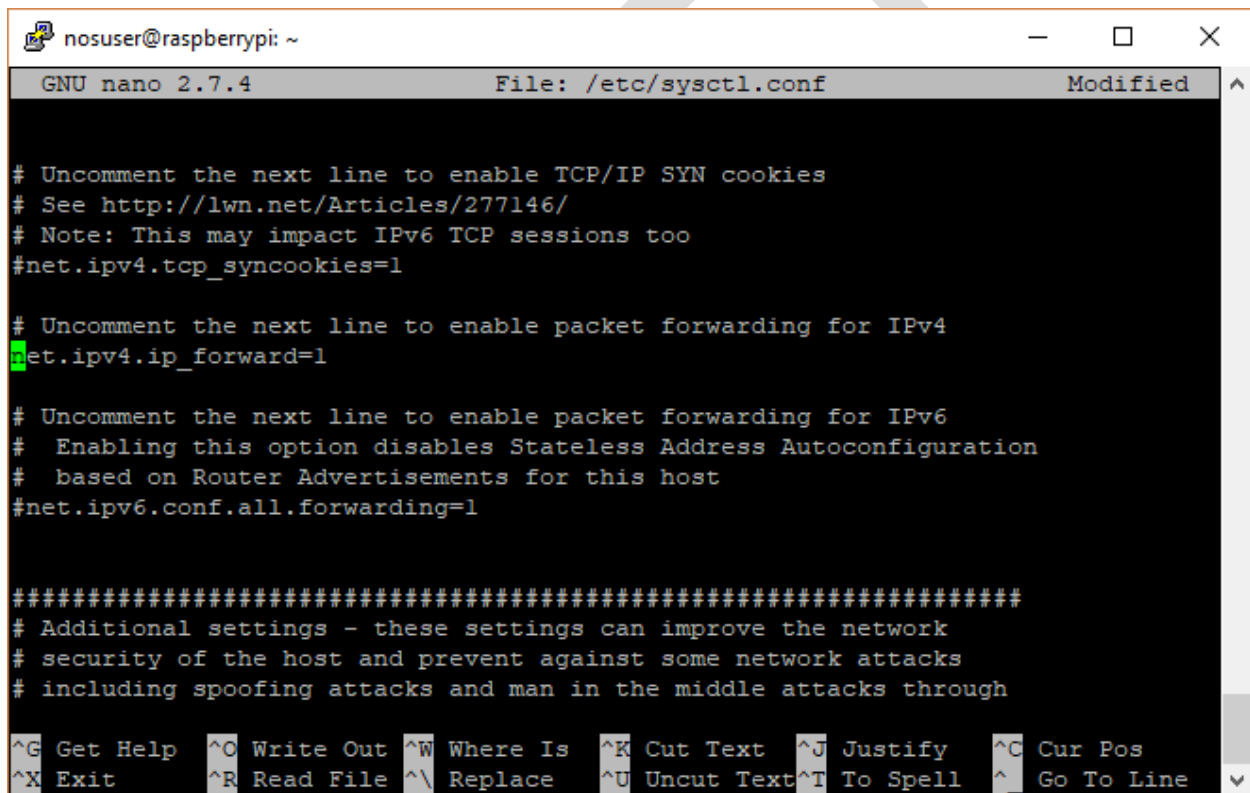
Once in the file, uncomment the following line (about line 28 or so):

```
#net.ipv4.ip_forward=1
```

to

```
net.ipv4.ip_forward=1
```

To save and exit, type Ctrl-X then answer with a y, then press enter.



```
nosuser@raspberrypi: ~
GNU nano 2.7.4      File: /etc/sysctl.conf      Modified
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

#####
# Additional settings - these settings can improve the network
# security of the host and prevent against some network attacks
# including spoofing attacks and man in the middle attacks through

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Now you need to re-read the file you just modified to set the ip_forward flag and echo what has been changed

Type the following command:

```
sudo sysctl -p
```

Preparing for and getting the JNOS source code

Now you need to create a folder to hold the actual JNOS source code.

Type the following commands:

```
cd /home/nosuser  
mkdir jnos2  
cd jnos2
```

Use rsync to get the code from the web site.

Please note that there is a trailing period in the below command, and that trailing period is preceded by a space. This space and dot must be present as written, or funny and strange things may happen, or simply not happen at all. You have been warned. I have found that sometimes when you copy/paste the command from this document you get an error. Try typing it in as shown; that always worked in those cases.

Type the following commands:

```
rsync -a www.langelaar.net::jnos2 .
```

What this command does is copy all of the associated files and directories into the /home/nosuser/jnos2 directory. Further configuration will be needed later to customize what you have, but we will deal with that later.

Verify that the files downloaded by doing an "ls -la".

Type the command:

```
ls -la
```

Compiling JNOS

Before we get much further along, let's do a test compile to make sure the source is sane and buildable.

Type the following commands:

```
cd /home/nosuser/jnos2  
make
```

What you should now see is a bunch of compiler output as the program is compiled. There will be *warnings* and *notes*, but there should be no *errors*.

If it compiles successfully (finishes with no errors), start the jnos executable. Because it has no autoexec to follow it won't do much. We are just looking for the jnos> prompt. If we see it, good.

Type the following command:

```
./jnos
```

Once you get the jnos> prompt, go ahead and exit the program.

At the jnos> prompt, type:

```
exit
```

At the confirm prompt, type:

```
y
```

The jnos binary has a lot of unnecessary debugging symbols added, so when we get a good build, we strip them off.

Type the following command:

```
strip jnos
```

What this command does is remove the debugging symbols from the program. You do not normally need them, so to make the executable smaller, we "strip" them out.

Setting up a serial port

In order to use the Pi and JNOS, we need a serial port to attach a TNC or other KISS modem to.

A USB-serial adapter fed TNC will use a ttyUSB<X> device (e.g., ttyUSB0). A TNC-Pi will use a ttyAMA<X> device (e.g., ttyAMA0). More recent docs for the TNC-Pi suggest /dev/serial0. I'll pursue using a USB-Serial device.

Plug in your USB Serial device. Any USB port on the Pi will do. After a maybe 30 seconds or so, the Pi will have found the USB-Serial device and assigned it a device id.

Type the command:

```
dmesg | grep tty
```

You should find lines similar to:

```
[ 0.887947] 3f201000.serial: ttyAMA0 at MMIO 0x3f201000 (irq = 87, base_baud = 0) is a PL011 rev2
[ 0.900641] 3f215040.serial: ttyS0 at MMIO 0x0 (irq = 166, base_baud = 31250000) is a 16550
[ 9.418562] usb 1-1.2: pl2303 converter now attached to ttyUSB0
```

The last one line shown gives me the proper serial USB port to use, which in this case is ttyUSB0.

Install JNOS

There is an installer program made when you compiled the JNOS source, but not all of the pieces are in the right place. What follows is a sample session; the places that require an answer other than just pressing the Enter key are in **green**. Please use your own call, not mine, and choose a different password.

Type the following commands:

```
cd /home/nosuser/jnos2  
cp jnosinstaller installerv2.1/  
cp jnos installerv2.1  
cd installerv2.1  
sudo ./jnosinstaller
```

```
Welcome to version 2.2 of the JNOS 2.0 installer program  
(C)copyright 2005-2015 by Maiko Langelaar / VE4KLM
```

```
Just answer the questions as they pop up. IF you do not know how to answer  
a particular question, just hit the ENTER key to pick the default choice.
```

```
Enter the word 'install' to begin, or anything else to quit.
```

```
--> install
```

```
Enter the JNOS root directory (default is /jnos)
```

```
-->
```

```
using default value
```

```
<SNIP>
```

```
success!
```

```
Now we are going to create the autoexec.nos (configuration) file.
```

```
Enter your callsign
```

```
--> N8AVX
```

```
do you want to add a (AXIP) wormhole (yes or no) ?
```

```
no
```

```
do you want to add a (AXUDP) wormhole (yes or no) ?
```

```
no
```

```
do you want to add a TNC to this system (yes or no) ?
```

```
yes
```

```
Enter the serial port for the TNC (default - ttyS0)
```

```
> ttyUSB0
```

```
Enter the baud rate for the serial port (default - 9600 baud)
```

```
>
```

```
using default value
```

```
do you want the TNC to beacon every 20 minutes (yes or no) ?
```

```
yes
```

```
<SNIP>
```

```
creating security file - give yourself a good password ...
```

```
--> floopydude
```

```
<SNIP>
```

```
The next section requires careful reading !
```

```
Hit the ENTER key to continue
```

```
-->
```

```
<SNIP>
```

```
IF you are not comfortable with the above, or you are running a SystemD style
```

JNOS on a Pi v4.3

linux distro (I have no script), then you can run JNOS manually as follows :

```
cd /jnos
./jnos -d /jnos
```

You now have a basic JNOS system available to play with ...

Once it's running, 'telnet 192.168.2.2' from the linux command prompt and login to JNOS with your callsign and the password you entered earlier on.

Warning : This configuration uses the TUN kernel module, and is specifically setup to use the 'tun0' interface. IF you are running OpenVPN or other software dependent on the TUN kernel module, running JNOS could mess things up for the other software, and/or vice versa. IF there is a possibility of this, then edit the autoexec.nos file, and change 'tun0' to an unused interface.

Note: The instructions above to telnet will not work unless the JNOS program is running. In addition, the Pi running Raspian is not a SysV system, so the instructions on how to set it up to automatically run at boot time will not work with a default Raspian install.

Now let's go and set the proper file permissions so you can edit the files and run the jnos binary.

```
cd /
sudo chown -R nosuser:nosuser /jnos
cd /jnos
chmod +x startnos
```

You can now start the JNOS process with the following command.

```
sudo ./startnos /dev/tty7
```

DON'T PANIC. What this command does is use the startnos script to start JNOS and use a defined tty port for the JNOS command console. If you wish, you can simply start JNOS while in the /jnos directory with ./jnos -d /jnos as given in the installer output. While it is running you will not get a command prompt back until you exit the JNOS process you started on tty7.

Press Alt-F7 to get to the JNOS command console.

Note: Do not use a tty that has a getty running on it. Confusing things happen. If you know what I'm talking about, good. If not, trust me; just use the above command as the Pi running Raspian does not have a getty running on tty7.

You now have a basic JNOS system that just needs to be connected to a KISS enabled device. For this setup, Alt-F1 gets you back to your Linux command prompt screen, and ALT-F7 gets you to the JNOS process. If you are running in a windowed environment (Gnome, KDE, whatever) your keystrokes will be different. I don't know what those would be.

Of course a radio on the other end of that KISS device would help.

Exit out of JNOS.

systemd configuration

We need to decide on what tty we will run. It should be one that does not have a getty that gets spawned on it. For what we have done here, one would be tty8.

Type the command:

```
sudo cp /lib/systemd/system/getty@.service /etc/systemd/system/jnos@.service
```

and edited jnos@.service to my situation.

Type the command:

```
sudo nano /etc/systemd/system/jnos@.service
```

Modified line:

```
ExecStart=/jnos/startnos
```

Added to the end of that file:

```
Alias=getty.target.wants/jnos@tty7.service
```

Save the file and exit nano.

The following command is to be typed all on one line. The document isn't wide enough to show it all on one line, so it makes a line break at the space in the middle of the entire thing. Using ellipses to replace sections of the command for illustrative purposes, we have "sudo (...) [jnos@.service](#) /etc/(...)service. Note that there is a space between the end of the first argument ([jnos@.service](#)) and the beginning of the second argument (/etc/systemd).

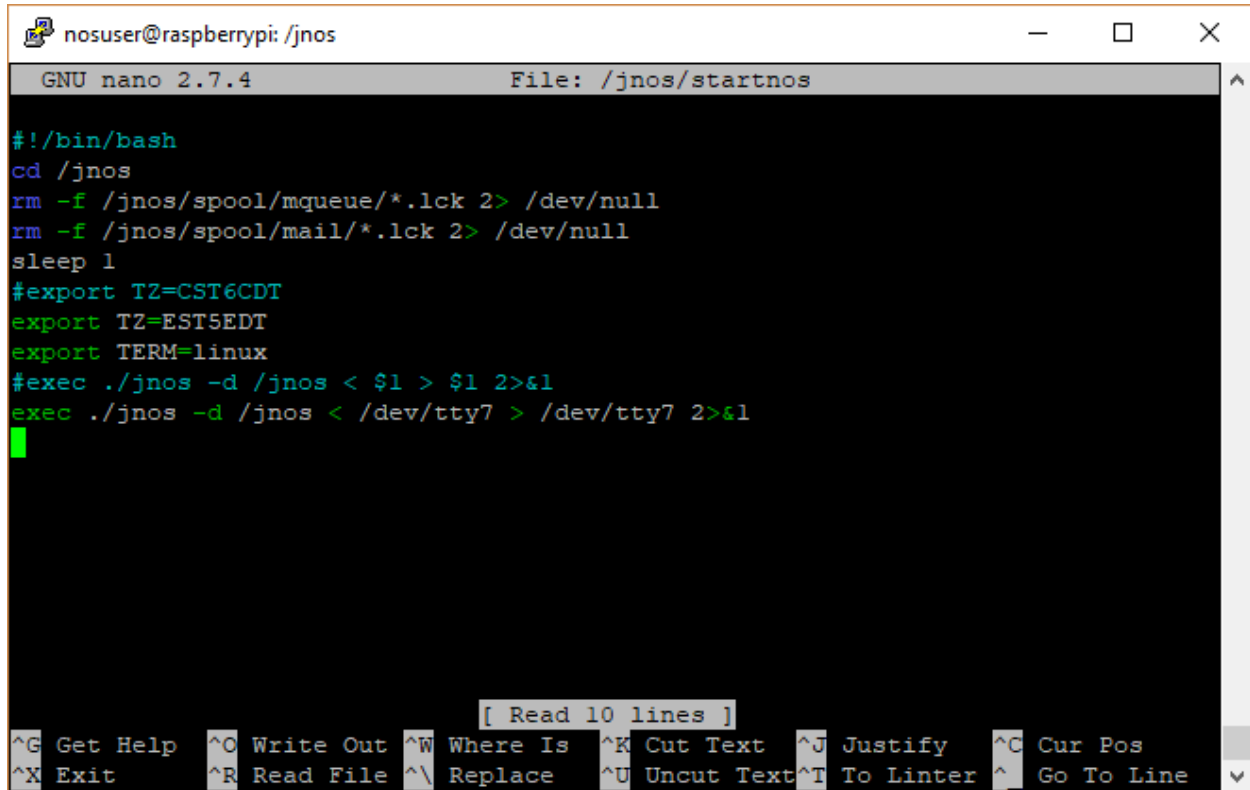
Type the following command all on one line:

```
sudo ln -s /etc/systemd/system/jnos@.service  
/etc/systemd/system/getty.target.wants/jnos@tty7.service
```

Now we need to modify the /jnos/startnos script to support use by systemd.

Type the command:

nano /jnos/startnos



```
nosuser@raspberrypi: /jnos
GNU nano 2.7.4 File: /jnos/startnos

#!/bin/bash
cd /jnos
rm -f /jnos/spool/mqueue/*.lck 2> /dev/null
rm -f /jnos/spool/mail/*.lck 2> /dev/null
sleep 1
#export TZ=CST6CDT
export TZ=EST5EDT
export TERM=linux
#exec ./jnos -d /jnos < $1 > $1 2>&1
exec ./jnos -d /jnos < /dev/tty7 > /dev/tty7 2>&1
```

[Read 10 lines]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line

Make the changes as shown in the screenshot above.

- Add #!/bin/bash to the top of the file
- Change TZ= to match your timezone
- Change the exec line to specify the tty device to be used. This must match what is used to set up systemd.

Ctrl-O and Ctrl-X to save and exit nano.

JNOS on a Pi v4.3

Start JNOS:

Type the command

sudo systemctl start jnos@tty7.service

On the keyboard, do Ctrl-F7 and you will see the JNOS process output.

Type exit at the jnos> prompt, it will ask for confirmation, then restart.

To stop it:

Type the command

sudo systemctl stop jnos@tty7.service

Of course, the next time the Pi reboots, JNOS will start.

Troubleshooting

/dev/ttyUSB0 is locked

```
BBS:
0 Command:
JNOS version 2.0k.2 (Linux)
by Maiko Langelaar (VE4KLM) - based on JNOS 1.11f, by James P. Dugal (N5KNX),
Johan. K. Reinalda (WG7J/PA3DIS), Brandon S. Allbery (KF8NH), and others.
Copyright 1991 by Phil Karn (KA9Q) and contributors.
/dev/ttyUSB0 is locked
Interface "vhf" unknown
input line: ifconfig vhf description "vhf - 1200 baud port"
Interface "vhf" unknown
input line: param vhf 2 256
Interface "vhf" unknown
input line: param vhf 3 1
Interface "vhf" unknown
input line: param vhf 5 1
Interface "vhf" unknown
input line: param vhf TxDelay 25
Interface "vhf" unknown
jnos> exit
Are you sure? y
JNOS Exiting - Uptime => 0:00:00:15
```

If you see something like this you need to manually remove the lock file found in /var/lock/.

```
nosuser@raspberrypi: /jnos $ ls /var/lock
asound.state.lock LCK..ttyUSB0 msubsys
nosuser@raspberrypi: /jnos  sudo rm /var/lock/LCK..ttyUSB0
nosuser@raspberrypi: /jnos  ls /var/lock
asound.state.lock subsys
nosuser@raspberrypi: /jnos
```

I can't use IP to connect with other systems

This is complicated.

To begin with, in order to use the encap routes to talk to other IP nodes on the 44.x.y.z network, you need a compatible IP address. The default autoexec.nos uses 192.168.2.2 for the IP address.

```
#
ip address 192.168.2.2
#
attach tun tun0 1500 0
#
ifconfig tun0 ipaddress 192.168.2.2
ifconfig tun0 netmask 255.255.255.0
ifconfig tun0 mtu 1500
#
shell ifconfig tun0 192.168.2.1 pointopoint 192.168.2.2 mtu 1500 up
#
```

That “ip address” line at the top needs to be a 44.x.y.z address to talk to other systems on the 44 AMPR net. You need an official IP address in that space. Please do not “roll your own”.

My autoexec.nos looks like this:

```
ip address 44.102.xxx.yyy (I'm not putting my IP in an example!)
#
attach tun tun0 1500 0
#
ifconfig tun0 ipaddress 192.168.1.227
ifconfig tun0 netmask 255.255.255.0
ifconfig tun0 mtu 1500
#
shell ifconfig tun0 192.168.1.226 pointopoint 192.168.1.227 mtu 1500 up
shell /usr/sbin/arp -i eth0 -Ds 192.168.1.227 eth0 pub
#
```

Here are some notes on what is in mine:

```
ifconfig tun0 ipaddress 192.168.1.227
```

This is the local IP address of the JNOS process. It is different from the IP address of the Pi, which in my case is 192.168.1.110. I use that IP to get to the Linux prompt on the Pi.

```
shell ifconfig tun0 192.168.1.226 pointopoint 192.168.1.227 mtu 1500 up
```

Here is where I set up the tunnel to connect JNOS to the outside world. The .226 address is simply an endpoint for the tunnel. It needs one, and you don't ever have to bother with it again. The .227 address is the other end of the tunnel in the outside world.

```
shell /usr/sbin/arp -i eth0 -Ds 192.168.1.227 eth0 pub
```

Here is where we complete the connection to the outside world. There are several ways to do this; this is how I choose to do it. Essentially, we are doing sort of a proxy ARP, causing the eth0 interface to respond to ARP requests for 192.168.1.227. This means any telnet sessions on the LAN to the .227 IP will connect directly to the JNOS process.

This still isn't a real answer, but like I said, it's complicated and will be covered in another document I'm working on. In the meantime, ask around and I'm sure someone can help if you really want to go further. This doc has gotten long enough as it is

Jim N8AVX