iPaq Intro, Python, and Connectivity

Feb 13, 2006
Larry Rudolph

Administration

- iPaq's and Mobile Phones very similar
  - both use python, bluetooth, internet
- This week:
  - Ipaq comments, Python, Network
    - Problem set, due in one week
- On your own, watch:
  - Jamey Hicks on "Linux on iPaq" (video)

Setting up iPaq

- Why Linux?
- Why not Linux on newer models?
- Things particular to these iPaqs
  - batteries
  - no solid connections (check cables often)
  - when in doubt, reinstall

Connecting to iPaq

- Serial cable (not usb)
  - after boot, can just login via some terminal program (minicom/hyperterm)
- ssh over the network
  - setup wireless connection to network
  - 'ssh -l root xxx.xxx.xx.xxx'
  - need ip address (do not need dns name)
  - make sure you are connecting to YOUR ipaq. Easy to mistype ip address.

Installing software

- /etc/ipkg.conf points to "feeds"
  - we will maintain our own feed
  - ipkg picks first matching file, not last while searching list of feeds
- Copy files to ipaq via
  - secure copy
    - "scp localFile.py root@ipaqip:/usr/bin/"
  - serial cable
    - xmodem/ymodem, sx on linux

Is your ipaq, your ipaq?

- Anonymous vs Personal handheld
  - Telephone example
    - Landline telephones are anonymous
    - Cell/Mobile phones are personal
- Tradeoffs
  - private state
    - can be lost or stolen; should be protected
    - setup overhead on user
    - daily underhead on user: setup once & forget
    - less dependent on connectivity
    - public/private keys easy to use once setup

Connectivity

- Ipaq: 802.11 (WiFi) or Bluetooth
- Mobile: GPRS (edge) or Bluetooth

Go to Python Tutorial Slides

Let's go through the slides from the Zope corporation on an introduction to Python by the inventor of python, Guido van Rossum

Server Code

```
import sys, socket

if len(sys.argv) < 2:
   print "usage: socketserver <port>"
   sys.exit(2)

# create the server socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

port = int(sys.argv[1])

# allow the socket to be re-used immediately after a close
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

s.bind( ("0.0.0.0", port) )

s.listen(5)             # start the server socket

(client, address) = s.accept()
print "accepted connection from %s:%d" % (address[0], address[1])

while True:
   data = client.recv(1024)
   if len(data) == 0:
      print "connection with %s closed." % address[0]
      break
   sys.stdout.write(data)
client.close()
```

Client Code

```
import sys
import socket

if len(sys.argv) < 3:
   print "usage: socketclient <address> <port>"
   sys.exit(2)

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.connect( (sys.argv[1], int(sys.argv[2]) ) )

print "connected.  type stuff."

while True:
   data = sys.stdin.readline()
   if len(data) == 0:
      print "closing connection with server"
      break

   s.send(data)
```

Online Tutorials

- Tutorials
  - http://www.python.org/doc/tut/tut.html
  - http://diveintopython.org/
  - http://www.intelinfo.com/newly_researched_free_training/Python.html
- use google or go to python.org

Discussion about network infrastructure

- Initialization
  - Network
    - Static IP, DNS server -- why IPv6 and why not
    - DHCP: get ip and dns server -- vast improvement
  - Servers

Let's design it right

- What do we want?
  - Everything should just work without setup
- Observation
  - most interaction is local
  - remote interaction is rare
  - overhead for rare cases is ok
- How to setup/find a server without a fixed name or ip address?
  - Interactive class discussion