**Objectives**

1. Research information about software for a specific operating system (OS) environment. You will be assigned one of the operating systems form the list below. You will also be provided with a list of topics to investigate.
2. Organize your rough research information into a list of topics, sub-topics and facts. This process will involve identifying sub-topics, rearranging your rough research notes, and selecting (or highlighting) interesting facts.
3. Report a summary of your research in the form of a “concept map”. Use the PowerPoint template provided as a starting point. The concept map should only include the best and most interesting information from your organized research notes.
4. Your concept map can be created using: Smart Ideas, Prezi, PowerPoint or other similar applications.

****

**Step 1 – Organized Research**

Research information about your assigned operating system (OS) environment.

* Guide your research according to the suggested topic list below
* Feel free to copy-and-paste as long as you keep track of your bibliographic references.
* Do not be too picky or concerned about formatting as you will organize this information later in step 2
* Select things that look interesting and don’t forget to include graphics images as well
* Upload your rough research notes to your repository when you are done.

Topic A – Application Software

The primary goal of QNX Neutrino is to deliver the open systems POSIX API in a robust, scalable form suitable for a wide range of systems — from tiny, resource-constrained embedded systems to high-end distributed computing environments. The OS supports several processor families, including x86, ARM, XScale, PowerPC, MIPS, and SH-4.

QNX is a mobile operating system that was originally developed for embedded systems. The operating system's developer, QNX Software Systems, was acquired by Research in Motion (RIM) and the OS adapted for use in the BlackBerry Playbook tablet.

QNX is a real time operating system originally developed by Canadian company QNX Software Systems, which was later acquired by BlackBerry. QNX stands for QNX is Not Unix.

Provide a summary of most important user application software targeted by this operating system and how it is similar to and deferent from standard PC software. Suggested sub-topics include:

* User (client) or network (server) applications
* Batch (run without user input) or interactive (user focused) processing
* Off-the-shelf (purchased) or custom developed applications
* Programming environment and languages supported

Topic B – Hardware

A BSP, or board support package, is the name given to the software responsible for hardware specific operations required to get a real-time operating system (RTOS) up and running. BlackBerry® QNX, with support from hardware and silicon partners, offers a broad and highly optimized level of hardware support for its software, including QNX® Software Development Platform 7.0 (QNX® SDP 7.0). QNX SDP 7.0 provides a 32-bit and 64-bit OS that builds on the proven reliability of BlackBerry QNX technology, and raises the bar for security and performance in mission critical applications. The QNX SDP 7.0 BSPs support a wide array of boards based on ARMv7, ARMv8, and x86 architectures. Source code is available for all QNX BSPs to allow modifications for your custom boards. Provide a summary of the hardware targeted by this operating system and how it is similar to and deferent from standard PC hardware.

Suggested sub-topics include:

* Speed of processors / memory
* Capacity of memory / attached disks
  + Description: The devb-ram driver creates a RAM disk interface. When the capacity option isn't specified, devb-ram creates a 2 MB RAM disk.
* Is it designed for home / office / corporate data center / industrial use
* Is it designed for client / server / network use

Topic C – User Interface

**Best-in-class support for HTML5**

The QNX HTML5 engine provides support for the HTML5 standard, and related standards and technologies such as CSS3, the JavaScript scripting language, plus associated standards, such as AJAX, JavaScript Object Notation (JSON), and XML. It also supports HTML5-specific features such as WebSocket, WebGL, session storage, offline applications, worker threads, DOM improvements, and the <canvas>, <audio>, and <video> elements. HTML5 applications can support a variety of user interface technologies, including interactive displays, audio, and video. With HTML5, developers can use a common tool set to build applications for QNX-based embedded devices, mobile devices, or applications to be hosted in the cloud.

**Multimedia support**

The QNX SDK for Apps and Media allows device manufacturers to rapidly integrate the latest multimedia capabilities customers want without worrying about the details of media access and management.

**Qt - Fully ported, integrated, and optimized for QNX**

Qt implementation with QNX leverages optimizations derived from integration with QNX OpenGL ES 2.0 library and graphic drivers to deliver best in class performance. You can mix and match Qt applications with other HMI technologies using the QNX composition manager architecture. Your applications can leverage a well-defined and optimized Qt <-> OS services layer through PPS (Persistent Publish Subscribe).

Provide a summary of the user interface and input devices targeted by this operating system and how it is similar to and deferent from a standard PC. Suggested sub-topics include:

* Does it support a windowed environment, command line, or network users
* Does it support multiple users at a time or single users
* Does it support multiple applications or a single application at a time
* Does it get rebooted (powered on / off) or is it always on

Topic D – Device Management

Provide a summary of the devices (disks, printers, etc.) and memory managed by this operating system and how it is similar to and deferent from a standard PC. Suggested sub-topics include:

* What types of disk drives and file systems does it support
* What type of input devices does it support
* What type of output devices does it support

**Floppy disks**

The driver for a floppy drive is devb-fdc. In order to use a floppy disk, you need to ensure that the floppy controller is enabled in the BIOS, and that the BIOS is configured to recognize the correct type of floppy drive (e.g. 1.44MB/2.88MB). The driver uses these locations as default:

I/O port 0x3f0

IRQ 6

DMA 2

If your controller is located at a different address, you can change these locations in the driver's options.

**Hard disks**

A self-hosted system, by default, detects the disk controller that's installed on the system, and then starts the appropriate driver for it.

On a self-hosted system, the diskboot utility in the OS image starts the block I/O drivers. If you want to change the way that the driver is started, you'll need to change the startup image and the options to diskboot

**EIDE**

EIDE interfaces use the devb-eide driver, which by default automatically detects the interface and devices attached to it. This driver includes support for UDMA (Ultra Direct Memory Access) modes, along with the generic PIO (Programmed Input/Output) modes. The supported hardware list includes adapters and their supported features; see the introduction to this chapter.

You can start the devb-eide driver without any options and, by default, it automatically detects the EIDE controller on the system

**SCSI devices**

A SCSI (Small Computer Systems Interface) bus is simply another bus that you can attach multiple peripherals to. Neutrino supports many brands and varieties of SCSI adapters; see the devb-\* (block-oriented) drivers in the Utilities Reference.

When the SCSI driver starts up, it scans the bus for attached devices. When the driver finds a supported device, it creates an entry in the /dev directory (e.g. a hard drive is hdx, where x is the number of the drive, starting from 0).

If the driver doesn't find any devices, it might not know the device ID of the adapter. Passing the device ID and vendor ID to the driver often corrects this problem. On a self-hosted system, you can pass these options to the driver via diskboot; see Controlling How Neutrino Starts.

In the following example, the driver automatically scans for SCSI devices on the chain and adds them into the /dev directory as they're found.

**SCSI RAID**

Currently, Neutrino supports only hardware RAID (Redundant Arrays of Independent Disks) devices. There are many third-party solutions for SCSI RAID available for Neutrino; search for them on the Internet.

**LS-120**

LS-120 is a SuperDisk drive that uses new technology to greatly improve head alignment, enabling a much greater storage capacity (120 MB) than conventional 3.5-inch disks. Neutrino treats an LS-120 drive like an EIDE drive.

**ORB**

An ORB drive is a fast, large-capacity, removable storage disk drive that uses 3.5″ storage media and attaches to the EIDE (ATA) chain. Ensure that the hardware is set up correctly and that the BIOS detects the hardware properly. An ORB drive is simple to set up, and appears in the /dev directory as a hard disk.

**RAM disks**

A RAM disk is a storage area that exists only in memory but looks like a hard disk. You can add one to your system by using devb-ram, but this is a RAM disk with the overhead of a block filesystem; by default, it's initialized and formatted for an fs-qnx4.so filesystem (unless you specify the ram nodinit option).

**Input devices**

The devi-\* set of drivers handles input under Photon. The type of input device attached to your system determines which driver you need to use. Photon input can consist of a single mouse, a mouse and a keyboard, or many mice at the same time (provided you have the space).

The inputtrap utility automatically detects basic input devices (non-USB keyboards and mice). The Photon startup script invokes this utility after starting the graphics adapters.

You can override the automatic detection by creating an input trap file, /etc/system/trap/input.hostname.

Topic E – Security

**General OS security**

It should be fairly obvious that security is important; you don't want someone to take control of a device and disrupt its normal functioning — imagine the havoc if someone could stop air traffic control systems or hospital equipment from functioning properly.

The importance of security to an individual machine depends on the context:

A machine behind a strong firewall is less vulnerable than one connected to a public network.

One that doesn't even have a network connection is in even less danger.

Part of securing a machine is identifying the level of risk. By classifying threats into categories, we can break down the issues and see which ones we need to concern ourselves with.

**Neutrino security in general**

Neutrino is a UNIX-style operating system, so almost all of the general UNIX security information (whether generic, Linux, BSD, etc.) applies to Neutrino as well. A quick Internet search for UNIX or Linux security will yield plenty of papers. You'll also find many titles at a bookstore or library.

We don't market Neutrino as being either more or less secure than other operating systems in its class. That is, we don't attempt to gain a security certification such as is required for certain specialized applications. However, we do conduct internal security audits of vulnerable programs to correct potential exploits.

For flexibility and familiarity, Neutrino uses the generic UNIX security model of user accounts and file permissions, which is generally sufficient for all our customers. In the embedded space, it's fairly easy to lock down a system to any degree without compromising operation. The ultrasecure systems that need certifications are generally servers, as opposed to embedded devices.

**Setting up a firewall**

Just as a building or vehicle uses specially constructed walls to prevent the spread of fire, so computer systems use firewalls to prevent or limit access to certain applications or systems and to protect systems from malicious attacks.

To create a firewall under Neutrino, you can use a combination of:

IP Filtering to control access to your machine

Network Address Translation (NAT) — known to Linux users as IP masquerading — to connect several computers through a common external interface

Provide a summary of the security features provided by this operating system and how it is similar to and deferent from a standard PC. Suggested sub-topics include:

* What types of user accounts and user permissions does it support
* How does it protect against conflicts / interference between legitimate application processes
* How does it protect against malicious software
* How does it support software updates and security updates

**Establish Security Culture**

For embedded systems, functional safety and cyber security must be tightly aligned. Safety engineering is focused on unintentional errors while security engineering focuses on deliberate malicious behaviours. Understanding the nuanced, and sometimes conflicting, interactions between these highly specialized engineering functions is the key to establishing a strong security culture.

**Establish Security Culture**

For embedded systems, functional safety and cyber security must be tightly aligned. Safety engineering is focused on unintentional errors while security engineering focuses on deliberate malicious behaviours. Understanding the nuanced, and sometimes conflicting, interactions between these highly specialized engineering functions is the key to establishing a strong security culture.

**Remediate and Protect Your Embedded Assets**

A meaningful software security plan must include performance scorecards for the supply chain and the SDLC.

BlackBerry QNX experts help prioritize areas of greatest risk, utilizing advanced tools for security design, and leverage best-in-class security solutions for embedded systems. We collaborate with our customers to build sustainable teams, processes and infrastructure.

BlackBerry QNX investigates all reports of security vulnerabilities affecting supported products and services. A security advisory is issued once the investigation is complete and the software update is released. Installing the recommended update(s) in this advisory will help maintain the security of your BlackBerry QNX product(s).

Topic F – Network Connectivity

Provide a summary of the network connectivity provided by this operating system and how it is similar to and deferent from a standard PC. Suggested sub-topics include:

* Is the computer stand-alone or part of a larger network
* What type of network and internet connections does it provide
* Does it provide other services such as backup, firewall, etc.

**Network manager (io-pkt\*)**

The io-pkt\* component is the active executable within the network subsystem. Acting as a kind of packet redirector/multiplexer, io-pkt\* is responsible for loading protocol and driver modules based on the configuration given to it on its command line (or via the mount command after it's started). Employing a zero-copy architecture, the io-pkt\* executable efficiently loads multiple networking protocols or drivers (e.g. lsm-qnet.so) on the fly— these modules are shared objects that install into io-pkt\*. At the bottom layer are drivers that provide the mechanism for passing data to and receiving data from the hardware. The drivers hook into a multi-threaded layer-2 component (that also provides fast forwarding and bridging capability) that ties them together and provides a unified interface for directing packets into the protocol-processing components of the stack. This includes, for example, handling individual IP and upper-layer protocols such as TCP and UDP. In Neutrino, a resource manager forms a layer on top of the stack. The resource manager acts as the message-passing intermediary between the stack and user applications. It provides a standardized type of interface involving open(), read(), write(), and ioctl() that uses a message stream to communicate with networking applications. Networking applications written by the user link with the socket library. The socket library converts the message-passing interface exposed by the stack into a standard BSD-style socket layer API, which is the standard for most networking code today.

**Using Wi-Fi with io-pkt**

When you're connecting to a Wireless Network in Neutrino, the first step that you need to do is to start the stack process with the appropriate driver for the installed hardware. For information on the available drivers, see the devnp-\* entries in the Utilities Reference. For this example, we'll use the driver for network adapters using the RAL chipset, devnp-ral.so. After a default installation, all driver binaries are installed under the staging directory /cpu/lib/dll.

**TDP and io-pkt**

Transparent Distributed Processing (also known as Qnet) functions the same under the old io-net and new io-pkt infrastructures, and the packet format and protocol remain the same. For both io-net and io-pkt, Qnet is just another protocol (like TCP/IP) that transmits and receives packets. The Qnet module in Core Networking is now a loadable shared module, lsm-qnet.so. We support only the l4\_lw\_lite variant; we no longer support the qnet-compat variant that was compatible with Neutrino 6.2.1.

**Using TDP over IP**

TDP supports two modes of communications: one directly over Ethernet, and one over IP. The “straight to Ethernet” L4 layer is faster and more dynamic than the IP layer, but it isn't possible to route TDP packets out of a single layer-2 domain.

At the driver layer, there are interfaces for Ethernet traffic (used by all Ethernet drivers), and an interface into the stack for 802.11 management frames from wireless drivers. The hc variants of the stack also include a separate hardware crypto API that allows the stack to use a crypto offload engine when it's encrypting or decrypting data for secure links. You can load drivers (built as DLLs for dynamic linking and prefixed with devnp- for new-style drivers, and devn- for legacy drivers) into the stack using the -d option to io-pkt. APIs providing connection into the data flow at either the Ethernet or IP layer allow protocols to coexist within the stack process. Protocols (such as Qnet) are also built as DLLs. A protocol links directly into either the IP or Ethernet layer and runs within the stack context. They're prefixed with lsm (loadable shared module) and you load them into the stack using the -p option. The tcpip protocol (-ptcpip) is a special option that the stack recognizes, but doesn't link a protocol module for (since the IP stack is already built in). You still use the -ptcpip option to pass additional parameters to the stack that apply to the IP protocol layer (e.g. -ptcpip prefix=/alt to get the IP stack to register /alt/dev/socket as the name of its resource manager).A protocol requiring interaction from an application sitting outside of the stack process may include its own resource manager infrastructure (this is what Qnet does) to allow communication and configuration to occur.

**IP-EN converter**

In order to use the Ethernet interface, the TCP/IP stack needs the services of a converter module to add/remove the Ethernet header. As we'll see, this isolation of hardware specifics from the down producer allows for easy addition of future hardware types. It also allows for the insertion of filter modules between the down producer and the converter, or between the converter and the up producer. In this case, the IP-EN converter basically provides ARP (Address Resolution Protocol) services.

**Ethernet driver**

At the lowest level, there's an Ethernet driver that accepts Ethernet packets (generated by the IP module), and sends them out the hardware (and the reverse: it receives Ethernet packets from the hardware and gives them to the IP module).

Topic G – User Account

**What does a user account do?**

A user account associates a textual user name with a numeric user ID and group ID, a login password, a user's full name, a home directory, and a login shell. This data is stored in the /etc/passwd and /etc/shadow files, where it's accessed by login utilities as well as by other applications that need user-account information.

sers log in with a user name and password, starting a session under their user ID and group ID

users create their own login environments

applications determine the user name and account information relating to a user ID and group ID if they're defined in /etc/passwd and /etc/group (e.g. ls -l displays the names — not the IDs — of the user and group who own each file)

utilities and applications accept user names as input as an alternative to numeric user IDs

shells expand ~username paths into actual pathnames, based on users' home directory information stored in their accounts

Groups are used to convey similar permissions to groups of users on the system. Entries in /etc/passwd and /etc/group define group membership, while the group ID of a running program and the group ownership and permission settings of individual files and directories determine the file permission granted to a group member.

When you log in, you're in the group specified in /etc/passwd. You can switch to another of your groups by using the newgrp utility.

**User accounts vs user IDs: login, lookup, and permissions**

Once you've logged in, the numeric user ID of your running programs and system resources determines your programs' ability to access resources and perform operations, such as sending signals to other processes. Textual names are used only by utilities and applications that need to convert between names and numeric IDs.

**What happens when you log in?**

You typically start a session on the computer by logging in (see Logging In, Logging Out, and Shutting Down); the configuration of your account determines what happens then.

When you log in, the system creates a user session led by a process that runs under your user ID and default group ID, as determined from your account entry in /etc/passwd.

The user ID and group ID determine the permission the process has to access files and system resources. In addition, if the process creates any files and directories, they belong to that user and group. Each new process that you start inherits your user ID and group ID from its parent process. For more information about file permissions, see “File ownership and permissions” in Working with Files.

Topic H – Cool Features

**High Availability**

If a device driver, protocol stack, or application experiences a problem, it does not take other components down with it. The QNX Neutrino RTOS high availability manager can terminate and restore the faulting component in isolation — often in just a few milliseconds, and without a reboot.

**Step 2 – Concept Map**

Create a “concept map” as a final report of your organized research.

* Use the diagram in the introduction as a starting point.
* You should have six (6) first level topics from “Application Software”   
  to “Network Connectivity”
* Each first level topic should have at least three (3) sub-topics
* Each sub-topic should be supported by a number of facts / items of interest

Select the best and most interesting information from your organized research.

* Summarize and edit your information to fit on the concept map.

Upload your Research Notes and Concept Map to your GitHub Repository

* Your concept map can be created using: Smart Ideas, Prezi, PowerPoint or other   
  similar applications.
* Option1: Create and upload a PDF of your concept map
* Option2: Include a link to your Concept Map in your Student Questions
  + Make sure that your link is Sharable so Mr. Nestor can open your map

**Appendix A**

|  |  |  |
| --- | --- | --- |
| **Operating System** | **Student 1** | **Student 2** |
| Ubuntu  (Linux) |  |  |
| z/OS  (IBM) |  |  |
| Solaris  (Oracle) |  |  |
| HP-UX  (Hewlett Packard) |  |  |
| Windows NT  (Windows Server) |  |  |
| Red Hat Enterprise (IBM Summit) |  |  |
| QNX  (Blackberry) | Dinesh | Suhkman |
| VxWorks  (Wind River) |  |  |
| AOSP  (Android Alphabet) |  |  |
| Ubuntu  (Linux) |  |  |
| z/OS  (IBM) |  |  |
| Solaris  (Oracle) |  |  |
| HP-UX  (Hewlett Packard) |  |  |
| Windows NT  (Windows Server) |  |  |
| Red Hat Enterprise (IBM Summit) |  |  |
| QNX  (Blackberry) |  |  |
| VxWorks  (Wind River) |  |  |
| AOSP  (Android Alphabet) |  |  |