Siemens Industrial Workstation Configuration Notes

---------Part 3: Booting and Shutting Down the System---------

\*Bootstrapping is the standard term for “starting up a computer”. The operating system’s normal facilities are not available during the startup process so the computer must pull itself up by its bootstraps.

--Boot Time is a period of vulnerability. Errors in configuration, missing or unreliable equipment,

and damaged file systems are all issues that must be noticed

--Boot configuration is one of the first tasks of a System admin and is one of the most sensitive

--When the workstation turns on, it first executes boot code that is stored in ROM. This code figures out how load and start the kernel. The kernel probes the system’s hardware and then spawns the system’s init process: process number 1

\*In normal operation, the systems boot themselves independently and are then accessed remotely by administrators and users. Admins need a recovery tool they can use if the disk crashes or a configuration problem prevents the system from completing the normal boot process.

--Booting in Single User Mode, Recovery Mode, or Maintenance Mode

--This mode does not allow Network Operation and requires physical access to the computer

--Steps in the Booting Process

--Reading of the boot loader from the master boot record

--Loading and Initialization of the kernel

--Device Detection and Configuration

--Creation of Kernel processes

--Administrator intervention (single user mode)

--Execution of system startup scripts

\*Kernel Initialization

--Pathnames differ in workstations but are similar to /boot/vmlinuz

--Most systems implement a two-stage loading process

--First stage is that the system ROM loads a small boot program into memory from the disk. This program called the boot loader then arranges for the kernel to be loaded.

--The second stage is the kernel probes the system to learn how much RAM is available. Some of the kernel’s internal data structures are statically sized, so the kernel sets aside some memory for itself when it stats. The kernel prints out how much physical memory is available.

\*Hardware Configuration

--One of the kernel’s first chores is to scrutinize the machine’s environment to see what hardware is present. As it probes the various system buses and inventories the hardware, the kernel prints out a line of cryptic information about each device it finds.

--Usually loads the device drivers as independent kernel modules.

\*Creation of the Kernel Processes

--After basic initialization, the Kernel creates several processes in user space (Not created with normal fork command).

--The number of spontaneous processes varies but init is always PID 1.

--These is no visible PID 0 under Linux but Unix has sched as process 0

--These spontaneous processes can be identified by brackets around their name in the ps command call. They may also contain a /# where the number is which CPU it is running on.

--List of common kernel processes

--kjournald = commits fileystem journal updates to disk

--kswapd = swaps processes when physical memory is low

--ksoftlrqd = handles soft interrupts if they can’t be dealt with at context switch time

--khubd = Configures USB devices

--Only init is a user process. The other kernel processes are just part of the kernel dressed up to look like processes for scheduling

--Init deals with starting the needs such as accepting logins, and starting system daemons

\*Operator Intervention

--if the system is put in recovery mode, a command line flag is passed by kernel and notifies init

--Prompts for the root password in the case the recovery mode is entered

--You can execute many of the same commands in this way but sometimes only the root partition is mounted. You must mount other filesystem by hand to use programs that don’t live in /bin /sbin /etc

--In many single user systems, the file system root directory starts off being mounted read-only. To change this you will have to remount with the following

--mount –o rw, remount /

\*Execution of Startup Scripts

--When the system is ready to run its startup scripts, it is recognizably UNIX even though it doesn’t look like it. The startup scripts are just normal shell scripts that have been selected to run by init according to an algorithm

\*Boot process completion

--After the startup scripts have run, the system is fully operational. System daemons such as DNS and SMTP servers are accepting and servicing connections.

--Init continues to perform an important role after booting. Init defines one single user and several network enabled run levels that determine which of the system’s resources are enabled.

--------------Part 3.2 Booting PC’S-------------

\*Booting a Computer

--When a machine boots, it begins by executing code stored in ROMs with the exact location and nature of this code varying depending on the system. The code is typically firmware that knows how to use the devices connected to the machine, how to talk to the network on a basic level, and how to understand disk-based file systems.

--The initial boot code is generally called BIOS (Basic Input Output System) and is much more simplistic as compared to the firmware of a proprietary workstation. PC’s have several levels of BIOS: One for the machine itself, one for the video card, and one for the SCSI card if the system has one.

--The built in BIOS knows about some of the devices that live on the motherboard, typically the IDE and SATA controllers and disks, network interfaces, power and temperature meters, and system hardware.

--The BIOS allows for you to select which devices you want the system to try to boot from. You can also specify an order to boot

--Once the BIOS has figured out what device to boot from, it tries to read the first block of the device. This 512 byte segment is known as the master boot record or MBR. The MBR contains a program that tells the computer from which partition to load a secondary boot program, the “boot loader”. The default MBR contains a simple program that tells the computer to get its boot loader from the first partition on the disk . Some MBR’s allow for knowing how to deal with multiple OS’s and Kernels. This loader is responsible for loading the kernel.

-------------Part 3.3 GRUB: The Grand Unified Boot Loader---------------

\*GRUB

--Developed by the GNU project, GRUB is the default boot loader for most UNIX and Linux system with Intel processors. GRUB’s job is to choose a kernel from a previously assembled list and to load that kernel with options specified by the administrator.

--There are two branches of GRUB lineage: the original GRUB now called GRUB Legacy and the newer GRUB 2. The name of GRUB 2 is deceptive as GRUB releases only have numbers between 1 and 2.

--By default, GRUB reads its default boot configuration from /boot/grub/menu.lst or /boot/grub/grub.conf. GRUB reads the configuration file at startup time so it allows dynamic changes at each system boot.

--GRUB also supports a powerful command-line interface as well as facilities for editing configuration file entries on the fly. From the command line, you can boot into other operating systems.

--When updates occur, Linux saves the old kernel so that it can be booted from if an issue arises from the new updated system.

--Systems can be set up to have more than one operating system. Each disk partition can have its own second-stage boot loader but the boot disk can have only one MBR. When setting up multiboot configuration, you must set which boot loader is the master. Always use GRUB over the windows based to be master.

\*Working with Startup Scripts

--After you exit from single-user mode or the standard boot sequence when it would have run, init executes the system startup scripts. These scripts are really just shell scripts that are interpreted by sh or bash.

--Scripts are kept in /etc/init.d. These startup scripts both start and stop services so this architecture also allows the system to be shut down in an orderly manner

--Common Tasks of the Startup Scripts

--Setting the name of the computer

--Setting the time zone

--Checking the disks with fsck

--Mounting the system’s disks

--Removing old files from the /tmp directory

--Configuring network interfaces

--Starting up daemons and network services

--Startup scripts print a description of everything they are doing. Administrators should not modify startup scripts.

--init scripts are used to some degree by all operating systems. Ubuntu uses Upstart which is very close to init and is basically the same thing

\*Init and its run levels

--Init is the first process to run after the system boots, and is the most important system daemon and is the ancestor of all other processes.

--Init defines at least seven run levels each of which represents a particular complement of services that the system that should be running.

--Level 0: The system is shut down

--Levels 1 and S: Represent single user mode

--Levels 2 through 5: Support for Networking

--Level 6: Reboot Level

--Levels 0 and 6 are special as the system can’t actually remain in them, it shuts down or restarts as a side effect of entering them.

--S Run level was created to address the need for having to enter the root password when entering single user mode. In Linux, it is only used to prompt for the password and is not a destination in itself

--Linux supports up to 10 run levels but most of these levels remain undefined.

--The /etc/inittab file tells init what to do at each run level. The telinit command changes intit’s run level once the system is up.

--Most Linux Distributions boot to run level 5 which involves the need for window server.

\*Overview of Startup Scripts

--The master copies of the startup scripts live in the /etc/init.d directory with each script being responsible for one daemon or one particular aspect of the system.

----------Part 3.7 Rebooting and Shutting Down---------

\*Rebooting and Shutting Down

--Traditional UNIX and Linux machines were very touchy about how they were shut down. Modern systems have become less sensitive with robust file systems.

--The shutdown command is the safest most considerate and most thorough way to initiate a halt or reboot or to return the system to single user mode. Most vendors have tampered with the shutdown command