Presentation #: 5045
Title: Virtual Administration of HP-UX Systems
Author: Mason C. Gibson
Intrinsix Corporation
4514 Chamblee-Dunwoody Road, Suite 283
Atlanta, GA 30338
Telephone 770-578-8038

Of all the jobs associated with computers, system administration is the least understood, and the most disorganized. Now, what do I mean by disorganized? Each hardware manufacturer, for its operating system, recommends daily tasks, monthly tasks, and even some yearly tasks, that should be performed in the operating system itself. There are also third party books to give you even more tasks that you should be doing in the operating system. For example, if you were interested in performance, there are certainly books on better performance, and how to retune your kernel. If you are interested in security, there are books on things you should be looking for in order to make your system more secure.

TASKS, TASKS, AND MORE TASKS

There are a lot of tasks associated with system administration. These tasks are outside of applications and hardware and the like. So you would think that a system administrator would have a list of things that should be done daily, weekly, monthly, and once a year. But in reality, I have never met a system administrator who had such a list. An administrator's time during the day is occupied by reacting to problems. Even if he had such a list, the problems would take priority over the list.

If you were to sit down and list all of the tasks that an administrator is supposed to do (even ignoring the application tasks--I am just looking at the operating system tasks), there are certain characteristics of this list of tasks that would be obvious. One is, there are a lot of tasks--a lot of tasks! Second, these system administration tasks are repetitive. Some are more frequent than others. A system administrator rarely does anything only once. He may not replace a disk often, yet he will do it more than once in his career. Other tasks are so repetitive they are done almost daily. So if you look at this from an engineering point of view, you will find that 80% of the tasks are frequent, and lend themselves to automation. That is, if you were to have enough time, to give it enough thought, you might take each task, and write a program or script to monitor it, and even to carry out the task. The other 20% of the tasks do not lend themselves to any automation. These are the tasks that make system administration the kind of job that administrators like. This 20% is challenging. It is interfacing with users. It is being helpful. It is using your skills to solve problems that crop up due to hardware problems, or just glitches of all kinds. When you look at the number of tasks, it becomes clear as to why we have to ignore the repetitive tasks and work on a reactive basis. The tasks are simply too many and too frequent to be handled by any organization, much less by one person.

This 80% is important. If the routine 80% of the tasks can be handled, then the system administrator might have fewer problems to react to. For example, we all know of cases of severe downtime problems, simply because something was ignored that should not have been ignored. Something that everybody knows they should do, often causes a problem, simply because it isn't done. This 80% is what I want to concentrate on, by introducing the term "Virtual Administration". **
While the term, "Virtual Administration", is new, the concept of virtual administration is as old as Unix itself. Back in the good old days, administrators recognized that there were things that could be automated. For example, everybody had a script that monitored disk space or filesystem space. As time permitted, administrators would write more of these scripts, to monitor more resources, to make

sure that they did not run out of resources, which might cause the system to crash. When these system administrators gathered in user groups and on other occasions, they would exchange these scripts. We all recognized the value of having more scripts. We knew we should take the time to write a few more, but never really had the time.

Virtual Administration is automating this 80% of the tasks of system administration. Virtual Administration has nothing to do with the other 20%. So if, somehow, Virtual Administration was done--that is, if an administrator could write enough scripts, and was knowledgeable enough to write such scripts--then this 80% could be automated and 20% would now be left for the administrator to do. Now, what I want to look at is a representative sample of these tasks in the 80%.

PROCESS ADMINISTRATION

Let's look first at process administration. Here, we should be concerned with the daemons. If a daemon goes down, something is not going to work. Whether or not the problem manifests itself in such a way that it is recognizable, the problem exists because a daemon is down. Therefore, daemons must be monitored. One thing that you can do is take a list of the daemons. In other words, if you do a ps -ef command, any daemon associated with a PID of 1 would be a candidate for monitoring. If it had a PPID of 1, and was not associated with a particular tty, in other words it had a question mark (?) for the tty number, this would be a candidate for monitoring. If you made such a list, and then at least daily make another list, and compare that list against the one you have stored, you will know if the necessary daemons are up and running. If they are not up and running, they need to be restarted.

Another area that we want to look at is runaway processes. A runaway process is defined as a process that is using more cpu time than it should. I define a runaway as using 50% or more of a cpu. In slow systems, this might be 40%, or even 30%. But the important thing is, systems do slow down during the day, and they can be slowed down by runaways. The way I sample runaways is to do a ps list, wait 4 minutes (240 seconds) and do another ps list. Then I compare the like PIDs, looking at the cpu time for those PIDs. If any process used more than 120 seconds, which is 50% of the time during the sample, then it is a runaway. The runaway should be investigated. It is very important to know why it exists. It is obviously affecting the system dramatically. There could be more than one. I have seen runaways exist naturally in some applications. For example, an invoice posting operation may be a runaway, simply because of the way the application was written. The company using that application may be aware that their system slows down in the afternoon, but they may not have any idea that the invoice posting program is the cause of the slowdown. If a runaway is a legitimate and necessary program, the system administrator needs to look into whether this process can be run at a better time, such as late in the day, first thing in the morning, or during lunch.

If the runaway is created by a process hanging up, maybe when somebody logged off, or got knocked off the network and it was left running, then it just needs to be killed. Regardless of the cause, vigilance is necessary to look for runaways on the system. Knowing about runaways enables the system administrator to handle them.

Another type of process to check is the orphan process. The orphan process is a process owned by a regular user, not a superuser, when that regular user is no longer logged into the system. Now there can be natural orphan processes. For example, if you are running a database, like Oracle, all the processes could be owned by a user named oracle, but that is a system user, and nobody ever logs in as oracle. Therefore, all processes owned by the user oracle appear to be orphans but are not. Some databases are started by the first user logging in after the system has been booted. Once the user logs off, the databases are still running under that user name. Therefore it looks like those database

processes are orphans, but they are not. These particular processes need to be defined as exceptions, so they will not be considered orphans. But there are many applications--more than one would think--that leave orphan processes out there that are not legitimate. For some reason, the user logs out, or is knocked off the network, or in some other way terminates abnormally, and there are processes left running that were started when the user logged in, which never have an opportunity to go away. An orphan process can be benign, in that it uses no cpu time. One would think that this would not be a problem. But consider that if your system is not shut down very often, and every orphan process occupies a position in the process table, how much time has to go by, before your process table becomes significantly filled by orphan processes, putting your normal processes in jeopardy because the process table may overflow? So orphan processes, even though they are not using cpu time, can pose a problem just by the fact that they occupy resources. Other orphans tend to become runaways. If a process was associated with some kind of I/O and it is no longer associated with a terminal, it begins to loop, then becomes a runaway. In either case, orphan processes should be monitored, because they are just troublemakers.

Another process is called a “hog”. A hog has many similarities with a runaway. It does not use enough cpu time to become a runaway, but it does use a significant amount. Even though it is not a runaway, one or two hogs can make a definite impact on a system. You need to do a process list--here you could use a ps -el command--which lists out cpu time. It would be interesting to put the highest user of cpu time at the top of the list, and sort it all the way down to the lowest user of cpu time. Any time that you do this, you will find that the 10 at the top of the list occupy almost all of the cpu time. Sometimes it is very surprising to see what is at the top of the list. I like to make such a list every hour, and keep 24 lists going all the time. By doing this I can see what the process table looked like every hour on the hour for the past 24 hours, enabling me to see which processes come and go, or which continue to use resources at a high rate.

FILES ADMINISTRATION

Next, let’s look at files administration. Here we need to look at garbage files, defined to be those which once had a good use but which no longer are needed. All applications leave such garbage files behind. Something just doesn’t happen like it should, therefore files are left long past their time of need. You are aware of the /tmp and other such temporary directories that somehow just collect garbage files. It is important to “take out the garbage” on a regular basis. For example, you need to remove “core” files from all your local filesystems . I like to run find lists, to find anything in the tmp areas that is over 3 days old and remove it. You might decide to remove it only if it is over 30 days old, but the longer you make it the more garbage you are going to accumulate. I like to remove nohup.out, dead.letter, anything associated with an application, anything associated with stream errors, anything associated with system crashes or dumps, and anything associated with uucp. These things must be sought out with a find command, and with appropriate age should be removed from the system.

The next type of file we want to look at is trash files, which are files that never had a use on the system. Trash files are characterized by the fact that the name of the file contains unprintable characters. If you have users who can create their own files, through word processors, spreadsheets or similar applications, they can create a file with a name like “m-a-left arrow-left arrow-N-right arrow-right arrow”. These files cannot be of any use, because they cannot be accessed, due to the nonprintable characters in their name. We always recognize that one might exist because when we do an ls command, the columns do not line up, and we think “Uh-oh, something is wrong here.” Trash files should be found and removed from the system. Finding these things is not an easy task. You have to get a list of all names on the filesystem and go through each name to determine if the ASCII characters in the name are appropriate.

Another type of file would be the mail files. Users have many opportunities to abuse a system. One way is to never get rid of their mail, so the mailboxes grow in size. What I like to do is take the total size of all mailbox files, then determine what percentage of that space is used by each user's mailbox. Then I can say to the user, "Mason, you are using 50% of the space. You need to clean out your mail, because you are using an unreasonable amount of space relative to all the other users."

Another area of concern is the system logs. There may also be application logs. System logs can grow to infinity. The author who created these system logs decided that the information in them was important, and that someone should look at them every once in a while and clean them out. It never happens! What I like to do is get a list of these files on my system and trim them to 200 records, on a weekly basis. If you don't do something, their infinite growth will get you someday.

Another type of file is the orphan file, which is owned by a user who is no longer a valid user on the system. You should be aware of these files. Either they need to be moved, given to other users on the system, or removed. If you run a list of the files in every directory, and compare their owners with /etc/passwd, if the owner is not in that file as a user, then it is an orphan file. If you don't remove the files, or give them to another user, you should at least change the owner of the files to root, so there won't be an orphan. You should also examine the files to be sure they do not have any significant permissions, such as "set user ID" or "set group ID". If they do, strip them. At least then they will be harmless. An orphan file can be assigned by any user to himself, and therefore may be a security breach. Orphan files just should not exist on a clean operating system.

Relating to files, one of the things I like to do is "checksum" the system directories. Applications can also have system directories, which contain only programs, no changing data. You can use the sum command to get checksums on all the files in that directory and store it. Sometime later you could do the same thing, then "diff" the two files to see if anything has changed. This would be a good idea for security. It is also a good idea because when you have lost the hard disk, put in a new one and reloaded the system, and it's 3 o'clock in the morning and you are ready to go home, if you could run those checksums on those directories you could have the peace of mind of knowing that all those directories are back the way they were. Checksumming directories is a useful, as well as necessary, system administration function.

Let's talk about missing files. If you ran a list of all the files on your system, and occasionally ran another list and compared them to see what had disappeared, you would know if you had any missing files. For example, I like to make my list every Sunday, and every day run the list again to find out what is disappearing each day. If you are not using any sophisticated backup methodologies, the list can be useful for determining when a file disappeared, and therefore what tape you need to retrieve the file from. Another extremely important use for this list would be, again, when the system has crashed and you have put it back together, it would be nice to run the list again to compare it to the list made the previous Sunday to be sure that nothing but trashy type files are listed as missing--no program files, no data files, no application files, no configuration, no devices are missing. It is just a good handle on all the files on your system, to have the list and to make the missing files list daily.

Users can create files, if they have access to word processors, spreadsheets, and the like. Since they can create files, they can be abusive. You need to monitor user home directories by getting the size of the home directory (I use du command). I like to add up the total size of all user home directories, then assign a percentage to the size of each user's home directory. I can then legitimately go to a user and say, "Look, you are using 60% of the total space that all users are using. Can't you get rid of some of these old documents or spreadsheets, and kind of clean house?"

SECURITY ADMINISTRATION

Security administration is a big concern these days to many companies. One of the things you need to have to monitor security is a list of all the world-writeable files. Use the find command to list all the files that have the "other" write permission. Once you look at this list, you may be surprised at the files that you thought were protected, that are not protected. Another list that you should have is made by going through the /etc/passwd file, noting the user group for each user. Then for each user group, use the find command to look for all files that are writeable by that group. Once you have this listing, you may be surprised again at what is there.

Backdoors are the earliest ways of getting into the Unix system. They should be monitored, with a program that goes through the /etc/passwd file looking for UID of 0. When you look at that list, it should contain only those that are known to you and authorized.

A Trojan horse is a file or program that currently exists and changes in such a way that it is of concern to the system administrator. For example, if I had a checksum of the login command, and that checksum changed, and I had not loaded any updates on the system, I would have to be concerned. Why is this login program different from the one that was there the last time I looked? There are other things related to Trojan horses. For example, if any .profile or any .login changes, or is deleted, I would be concerned. If a user creates a .rhosts in their own directory, this could be a doorway for someone to get into the system. You would want to know that this .rhosts file had been created. You certainly would want to know if any current .rhosts files changed in any way, in other words you would want a checksum on them. So, it is important to go through the system looking for files with "set user ID" and "set group ID" and checksum them. It is important to go through the system looking for things like .profile, .login, .rhosts and find out if they have changed. While you are at it, /etc/hosts.equiv and a host of other configuration files should also be checksummed. The ideal is to checksum anything that can be used as a Trojan horse.

Changes in file access are important. For example, if the group, owner or permissions of /etc/passwd were to change, you would like to know about it. There are lots of files that if access changes you would like to know about it. There are devices like /dev/kmem or /dev/mem, that if write permissions got onto those devices you would like to know about it. Changes in permissions on /etc/host and many more you would want to know about. To accomplish this file monitoring, you need to have a list of the files and you need to make a list of the file group, owner, and permissions. At least daily you need to make a new list and compare it with the previous list. You are certainly interested in any change in the checksum of a file in the system directory.

Logins with no password or logins with simple, easy to guess passwords are security breaches. Omission of users from security files, such as ftpusers, or leaving root out of ftpusers could be a severe security breach. You need to be aware that the file ftpusers exists, and which users, if any are in that file.

For good security, the passwd file needs to be clean. You don't need duplicate logins. Generally, you don't need duplicate UIDs, although some applications require them. At least you should be aware if any new ones are added.

PERFORMANCE ADMINISTRATION

Let's go into another area of great interest to system administrators: performance. Is the system running well, or not? In the future you will want to know if the system is running better or worse than it is today. One of the things you can do is write a benchmark program. You don't have to buy one, you can write one. Just make sure that you go burn some cpu time, write some tmp files out and erase them, write some more and erase them. What I like to do is write a benchmark program that runs about 30 seconds on my machine when there is no other activity on my system. Then I run that benchmark daily at a time when there is little or no other activity. I should always see results of about 30 seconds, give or take a second or two. In the future, if a user says a system is running slow, you can go back and look at the daily benchmarks and see if the system is running at a consistent speed. If so, the user must have been confused. Or if a vendor comes to you and says he can improve your system performance, you can install the product, then continue running the benchmark to see if performance improves. If it doesn't change, then you can go back to the vendor and ask "What was it that you were going to do for me?"

When you move from one operating system release level to another, you should note what it means to do that. For example, historically, any time you go up a release level in the operating system you lose some performance. The reason for this is that the operating system gets bigger, therefore more overhead. You need to know how much you lose when you go up. Sometimes when you have gone up one or two operating system levels, you have to compensate with more or faster hardware.

The other thing you can do to analyze performance on your system is to run the sar reports. If you could run the reports and if you could read the data, the data does tell you whether your system is cpu bound or disk bound, whether or not you have enough buffers, and whether or not you need more memory because you are swapping heavily. There are entire books out on how to read the sar reports.

The other thing that you need to look at are the static kernel tables that can be tuned--for example, the number of processes, the process table. If you get within 80% of the limit of a table, you should consider taking the system down and retuning the table limit higher. For example, if you run out of process table, you will get "unable to fork" messages, and processes will die quickly. If you run out of the file table, you will not be able to create pipes; you will not be able to open programs; you will not be able to write to files. You are just going to have a mess! These tables should be monitored frequently to be sure you are not approaching the limits, which would require retuning.

NETWORK ADMINISTRATION

Let's talk about another subject: network administration. There is a lot associated with this area, but certainly you need to look at hosts. Is the host available? In other words, ping the host occasionally to see if it is up or not. Some people may argue that if the host goes down you will know about it, because there will be a lot of reactive phone calls. But still, there are situations where monitoring the host will mean more than that phone call. You should also look at the interfaces. Particularly, you should look at the collision rate on an interface. Use netstat -i. The collision rate should not exceed 5%. If it does, that indicates a problem, either hardware or just slow interface. You should also look at the packet input error rate. Even 1% on this would indicate a very poor interface.

FILESYSTEM ADMINISTRATION

Now we will talk about some aspects of filesystem administration. Are my filesystems mounted? Do I have any free blocks? I like to monitor free blocks to make sure that I have 10% of the total blocks still free. If it falls below that I want to be told that I am below my threshold. The same thing with free inodes. I also like to monitor use of swap space, to make sure that I don't get above 80%.

SYSTEM FILES ADMINISTRATION

There are also administrative maintenance tasks to be done related to the system files. The passwd file needs to be scanned on occasion to make sure that every record in it has the right number of fields, that every user group appears in the group file. You need to check the group file. You need to make sure the passwd file doesn't have any duplicates, and that the shell is a viable shell. By the way, the shell should not be "set user ID". If it is, you need to know about that. The gettydef file can be routinely checked with the getty -c command. The inittab file is extremely important. You need to make sure that you don't have any duplicate labels in there, and of course that every record has the right number of fields.

VIRTUAL ADMINISTRATION

If you are overwhelmed at this time, by just the number of things that I have gone over, you should be! That was the intent. The reality is that there are so many tasks that should be done, and it would be great if they could be done, that no human or group of humans could adequately do them and still carry out the other 20% of system administration tasks. So these 80% need to be automated. The good news is--they can be and they have been.

What if you could have another system administrator, not a human one, but a virtual system administrator that actually resides in your computer, working 24 hours a day, 7 days a week, causing your system to handle all those repetitive tasks? It would handle the 80%, so that you can devote your time and talent to the remaining 20%. That would be "Virtual Administration".

The objectives of Virtual Administration are:

1. Do all the recommended maintenance tasks automatically, so the system administrator does not have to spend time writing and running scripts to do routine, repetitive tasks.

2. Do the recommended tasks in an orderly manner, on a regular schedule, so that important routine tasks are not pushed aside by more urgent crisis tasks.

3. Monitor system resources--daemons up or down, runaways, orphan processes, network problems, collision rate problems--all the things we have talked about and a great deal more.

4. Store data about the operation of the system. Unix is a data-less system. For example, you can't tell me how much space you had in a filesystem yesterday, much less how much was in there this time last month. Data is important. If you have a system problem, when you don't have data, you have to go ask a lot of questions of the user community. The information that you get, most of the time, is not reliable. Virtual Administration would keep and use logs of the areas of the system that I talked about and many more, so that you can know what is happening, and what has happened in the system. Rather than having to rely on users to know and remember what they and the system were doing at the time of the problem, it is better to go into the logs and see for yourself what was going on at the time of the problem, and before and after that. If Virtual Administration did nothing more than keep these logs, it would still be an extremely valuable tool for problem solving.

5. Bring standards to system administration. Since Virtual Administration is doing 80% of all the tasks associated with maintaining the system, the human system administrators can now follow the guidelines it sets up. For example, if you have two system administrators, and they want something to do done on boot, one administrator might put it in inittab, the other might put it in the rc structure or some other place. With Virtual Administration, there is one standard place to put a script to be executed during boot, and it is consistent across all platforms.

6. Provide a simple method to automate application-specific administration, maintenance or housekeeping tasks.

7. Notify a human if there is a potential problem that requires human attention. Virtual Administration is smart enough to solve many problems on its own. But some situations require human decision and intervention. In those cases, Virtual Administration should notify a system administrator or support provider. This notice can be provided on-site by beeping a terminal, or remotely by pager, fax, e-mail, or just about any way that you can think of.

The HP-UX (or Unix) operating system is powerful enough and intelligent enough to make Virtual Administration a possibility. It would require hundreds of programs and years of experience to make that possibility a reality.

"Virtual Administration" does not refer to just another tool to be used by a human system administrator to make his job easier. Virtual Administration is the automation of the routine 80% of the system administrator's job (the part that often does not get done at all), so that the system administrator can be free to do the 20% that only he can do. Virtual Administration would draw on the collective and cumulative knowledge of years of experience in system administration, and the recommendations of hardware manufacturers and others as to what should be done for optimum system administration.

Once you have a virtual system administrator on your machine, you would work with it for a while to make sure that it is identifying orphan processes correctly, for example. Work with it, in other words, like you would with any new system administrator. It is just going to carry out its job, 24 hours a day 7 days a week. All of these daily, weekly, monthly and yearly tasks will be carried out. Your system won't crash just because something was not done. Security violations and performance problems will be noted, and you will be made aware of them so that you can take appropriate action.

The old model of system administration pictures an overworked, underpaid system administrator frantically reacting to problems, and never having time to do the recommended routine and preventive tasks of system administration. With the new model of Virtual Administration, you have a very knowledgeable, qualified, and well-organized virtual system administrator operating on your system, taking care of the everyday tasks, maintaining the status quo and preventing problems, while the human system administrator interfaces with users, solves unexpected or unpreventable problems, and optimizes the usefulness of the system.

** NOTE: "Virtual Administration" is a general term used to describe the concept covered in this presentation. "Virtual Administrator™" is the trademarked name of a product belonging to Intrinsix Corporation, which implements this concept of Virtual Administration on HP-UX and other Unix-based platforms.