CONCOURS EXTERNE D'INGENIEUR D'ETUDES
BAP E – Ingénieur système et réseau (MI-27)

Ouvert au titre de 2008
Arrêté du 1er avril 2008 paru au JO du 5 avril 2008
Epreuve écrite
Notation sur 20 - Coefficient 3 – Durée 3h00

17 juin 2008 de 14h à 17h

Le candidat peut traiter les questions dans l'ordre de son choix.

Les copies seront corrigées anonymement. Veuillez donc respecter l'anonymat dans vos réponses et ne pas utiliser d'éléments permettant de vous identifier.

Une attention particulière sera portée à la qualité de la présentation, à l'orthographe et à la grammaire.

Connaissance du monde de la recherche

Exercice 1 (notation sur 1,5 point)

Quelles sont les spécificités d'un EPST ? Comment intégreriez vous ces spécificités dans le métier que vous aurez à exercer ? (Votre réponse devra faire entre 10 et 20 lignes)

Système

Exercice 2 (notation sur 1,5 point)

Qu'est ce qui différencie un processus standard d'un *daemon* système sur un système Unix/Linux ? Répondre en une dizaine de lignes.

Exercice 3 (notation sur 2 points)

Expliquez le principe et le fonctionnement de PAM.
Commentez le contenu du fichier ci-dessous :

cat /etc/pam.d/system-auth

```
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 500 quiet
auth        required      pam_deny.so

account     required      pam_unix.so
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 500 quiet
account     required      pam_permit.so

password    requisite     pam_cracklib.so try_first_pass retry=3
```

```
password    sufficient    pam_unix.so md5 shadow nis nullok try_first_pass use_authtok
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session     required      pam_unix.so
#
```

Exercice 4 (notation sur 1,5 point)

Décrivez les possibilités de "*Virtual Host*" implémentables sur un serveur WEB (de type apache, par exemple). Dans quel cas et pour quelle(s) raison(s) les utiliseriez-vous ?

Exercice 5 (notation sur 2 points)

Ecrire un script (dans l'un des trois langages: shell, perl ou python) permettant de journaliser quotidiennement les fichiers de trace d'un système Linux (`daemon`, `messages`, `secure`). Ces fichiers résident dans `/var/log` et sont conservés pendant une semaine. Les fichiers journalisés seront numérotés en les suffixant par un chiffre allant de 0 à 6, du plus récent au plus vieux.

Exercice 6 (notation sur 1,5 point)

Quelles sont les principales fonctionnalités d'un gestionnaire de version de code source (ou documents) ? Citez ceux que vous connaissez.

Connaissance de la langue anglaise

Exercice 7 (notation sur 1,5 point)

Que répondriez-vous (en **anglais**) à cet utilisateur anglophone ? Pour des raisons d'anonymat des copies, ne signez pas votre réponse.

```
Date: Tue, 02 May 2008 17:26:45 +0200
From: Tom Junherd <Tom.Junherd@inria.fr>
To: sysadmin@inria.fr
Subject: Computer crash

My computer didn't accept any command, so I restarted it.  Now the Linux
desktop is not loading properly after the logon.  After the background image is loaded, I
can move the mouse, but appart from that nothing happens.

What should I do to get it working again?

-- Tom Junherd
```

Exercice 8 (notation sur 2 points)

Vous trouverez en annexe un document en anglais sur le système Plan9.
Comparez en une quinzaine de lignes le système Plan9 et le système Unix. (Répondez en **français** à cette question)

Problème : Etude de cas (notation totale sur 6,5 points)

Vous travaillez à l'INRIA en tant qu'administrateur système dans le service des moyens informatiques. Vous appartenez à l'équipe système Unix.

Question P.1
Un projet de recherche travaillant dans le domaine de la cryptologie vient vous demander conseil pour l'acquisition d'un serveur de calcul. Une contrainte forte du projet est que la machine soit sous Linux.
Votre contact technique dans le projet a déjà fait une petite étude de marché et souhaite acquérir la configuration suivante:

- machine rackable 2U,
- double alimentation,
- processeur Xéon de 3Ghz,
- 2 Go de RAM,
- disques RAID (RAID1) de 146 Go.

Quelles questions poseriez-vous à ce correspondant technique pour vérifier que la configuration qu'il a choisie correspond à ses besoins ? Argumentez.

Question P.2
Les membres de ce projet de recherche veulent maintenant disposer d'un serveur web accessible depuis l'Internet. Ils expriment les besoins suivants :

- le serveur distribue des pages publiques et des pages réservées à des utilisateurs authentifiés,
- le serveur accède à une base de données afin de fournir des pages dynamiques,
- les chercheurs du projet doivent pouvoir ouvrir des sessions interactives pour gérer les pages ou administrer ce serveur.

L'ensemble de ces services tournera sur la même machine.

2.1) Quelle architecture logicielle leur proposeriez vous pour répondre à ces besoins ? Expliquez les interactions entre les différents éléments de cette architecture.
2.2) Quelles mesures techniques et organisationnelles prendriez-vous pour sécuriser ce serveur ? Présentez les en les regroupant par catégorie.

Question P.3
Après quelques mois d'utilisation, le projet s'aperçoit qu'il a largement sous estimé l'utilisation du serveur web ci-dessus. Il vous sollicite à nouveau pour proposer une solution plus robuste en terme de performances et de disponibilité du service. Vous avez carte blanche, que proposez-vous ? Argumentez votre réponse.

Annexe : description du système Plan9 (from Wikipedia, the free encyclopedia)

Plan 9 from Bell Labs is a distributed operating system, primarily used as a research vehicle. It was developed as the research successor to Unix by the Computing Sciences Research Center at Bell Labs between the mid-1980s and 2002. Plan 9 is most notable for representing all system interfaces, including those required for networking and the user-interface, through the filesystem rather than specialized interfaces. Plan 9 aims to provide users with a workstation-independent working environment through the use of the 9P protocols. Plan 9 continues to be used and developed in some circles as a research operating system and by hobbyists.

Overview

All resources as files

One of the key features adopted from Unix was the use of the file system to access resources. Before Unix, most operating systems had different mechanisms for accessing different types of devices. For example, the application programming interface (API) to access a disk drive was vastly different from the API used to send and receive data from a serial port, which in turn was different from the API used to send data to a printer. Unix attempted to remove these distinctions. All device drivers were required to support meaningful read and write operations as a means of control. This lets programmers use utilities like mv and cp to send data from one device to another without being aware of the underlying implementation details.

However, at the time, many key concepts (such as the control of process state) did not seem to map neatly onto files. As new features like Berkeley sockets and the X Window System were added, they were incorporated to exist outside the file system. New hardware features (such as the ability to eject a CD in software) also encouraged the use of hardware-specific control mechanisms like the ioctl system call.

The Plan 9 research project rejected these different approaches. Each Plan 9 program views all available resources, including networking and the user-interface resources (like the window it is running in), as part of a hierarchical file system, rather than specialized interfaces.[2]

Design concepts

Plan 9's designers were interested in goals similar to those of microkernels, but made different architecture and design choices to achieve them. Plan 9's design goals included:

* Resources as files: all resources are represented as files within a hierarchical file system
* Namespaces: the application view of the network is a single, coherent namespace that appears as a hierarchical file system but may represent physically separated (locally or remotely) resources
* Standard communication protocol: a standard protocol, called 9P, is used to access all resources, both local and remote

Filesystems, files, and names

Plan 9 extended the system beyond files to "names", that is, a unique path to any object whether it be a file, screen, user, or computer. All are handled using the existing Unix standards, but extended such that any object can be named and addressed (similar in concept to the more widely known URI system of the world wide web). In Unix, devices such as printers are represented by names using software converters in /dev, but these addressed only devices attached by hardware, and did not address networked devices. Under Plan 9 printers are as virtualized as files, and both can be accessed over the network from any

workstation.

Another Plan 9 innovation was the ability for users to have different names for the same "real world" objects. Each user could create a personalized environment by collecting various objects into their namespace. Unix has a similar concept in which users gain privileges by being copied from another user, but Plan 9 extends this to all objects. Users can easily spawn "clones" of themselves, modify them, and then remove them without affecting the resources from which they were created.

Union directories

Plan 9 also introduced the idea of union directories, directories that combine resources across different media or across a network, binding transparently to other directories. For example, another computer's /bin (applications) directory can be bound to one's own, and then this directory will hold both local and remote applications and the user can access both transparently. Unix links and filesystem mounts made the original directory disappear. Using the same system, under Plan 9 external devices and resources can be bound to /dev, making all devices network devices without additional code.

/proc

The /proc directory, in which all running processes are listed, illustrates how these features work together to produce a greater whole. This special Plan 9 "file system" has also been adopted by Linux and other later operating systems. Processes appear as named objects (sub-directories with info and control files) under /proc, along with other kernel resources, giving the user a dynamic I/O channel to send commands to them and read data from them. The user does not have to use a limited set and form of system calls to interact with the kernel from compiled programs; rather, he or she can use tools such as ls and cat to search, query and manipulate processes.

Users can also mount /proc directories (and any other special file systems) from any other machines into their namespace as well, interacting with them as if they are local. The result is a distributed computing environment assembled from separate machines -- terminals that sit on users' desks, file servers that store permanent data, and other servers that provide faster CPUs, user authentication, and network gateways, all using the existing hierarchical directory/name system familiar to most computer users. A user can "build" a system by collecting up directories on fileservers, applications running on servers, printers on the network and then bind them all together into their personal namespace running on a terminal.

/net

Plan 9 does not have system calls for the multitude of communication protocols or device driver interfaces. For example /net is the API for all TCP/IP, and it can be used even with scripts or shell tools, writing data to control files to write and read connections. Relevant sub-directories like /net/tcp and /net/udp are used to interface to prospective protocols. You can implement a NAT by mounting a /net, which is the Plan 9 TCP/IP API, from a perimeter machine with a public IP, while connecting to it from an internal network of private IP addresses, using the Plan 9 protocol 9P in the internal network. Or you can implement a VPN by mounting a /net directory from a remote gateway, using secured 9P over the public Internet.

Here would be an example of using union (a stack) directories in /net: just like inheritance in OOP, you can take one (possibly remote) /special directory and bind another local special directory on top of that, adding some new control files and hiding others. The union directory now is like a child object instance of the original parent. The functionality of the original can be partially modified. Consider the /net file system. If you modify or hide its /net/udp sub-directory you may control or extend the UDP interface with local filter processes, still leaving the original /net/tcp running intact, perhaps in a remote machine. Note that name space is per process: if you give an untrusted application a limited, modified /net union directory, you restrict its access to the net.

All this makes it easy to combine "objects" or file systems written in different languages on different systems, while using standard naming, access control and security of the file system, largely transparently to the programmer.

This is similar to the facility offered by the mount_portal[1] command in BSD which by convention is mounted on /p instead of /net with only /tcp available.

Networking and distributed computing

Plan 9 is based on UNIX but was developed to demonstrate the concept of making communication the central function of the computing system. All system resources are named and accessed as if they were files and multiple views of the distributed system can be defined dynamically for each program running on a particular machine. This approach improves generality and modularity of application design by encouraging servers that hold any information to appear to users and to applications just like collections of ordinary files.

Key to supporting the network transparency of Plan 9 was a new low-level networking protocol known as 9P. The 9P protocol and its implementation connected named network objects and presented a file-like system interface. 9P is a fast byte-oriented (rather than block-oriented) distributed file system that can virtualize any object, not only those presented by an NFS server on a remote machine. The protocol is used to refer to and communicate with processes, programs, and data, including both the user interface and the network. With the release of the 4th edition, it was modified and renamed 9P2000.

Impact

Plan 9 demonstrated that a central concept of Unix -- that every system interface could be represented as sets of files -- could be implemented and made functional in a modern distributed system. Some ideas from Plan 9 have been implemented in other operating systems. Unix-like operating systems such as Linux have implemented some of Plan 9's file system, the UTF-8 character encoding, and limited forms of rfork-like system calls. Additionally, several of Plan 9's applications and tools, including the rc shell, have been ported to Unix and Linux systems and have achieved some level of popularity.

However, Plan 9 itself has never surpassed Unix in popularity, and remains primarily a research tool. Plan 9 has been criticized as "seem[ing] to function mainly as a device for generating interesting papers on operating-systems research." Eric S. Raymond in his book The Art of Unix Programming speculates on Plan 9's lack of acceptance:

> "Plan 9 failed simply because it fell short of being a compelling enough improvement on Unix to displace its ancestor. Compared to Plan 9, Unix creaks and clanks and has obvious rust spots, but it gets the job done well enough to hold its position. There is a lesson here for ambitious system architects: the most dangerous enemy of a better solution is an existing codebase that is just good enough."

Other critics of Plan 9 include those critical of Unix in general, where Plan 9 is considered the epitome of the "Worse is better" school of operating system design. Common criticisms include the relative lack of "polish" and development in Plan 9's windowing system and Plan 9's relative lack of maturity as a commercial-grade body of software.

Plan 9 proponents and developers claim that the problems hindering its adoption have been solved, and its original goals as a distributed system, development environment, and research platform have been met, and that it enjoys moderate but growing popularity. Inferno, through its hosted capabilities, has been a vehicle to bring Plan 9 technologies to other systems as part of heterogeneous computing grids.