LOOKING AFTER PEOPLE

This is perhaps a rather curious name for a chapter, considering that we have all the time been pushing the idea that the system is there to solely to serve the people who use it. So far, though, we've concentrated on what the machinery should be like to make it reasonably congenial for people to use; now we want to say something about the information about people which the system must have in order to work in the desired way.

WHY DOES AN OPERATING SYSTEM NEED TO WORRY ABOUT PEOPLE ?

Except for the obvious interface requirements which we've already discussed, it doesn't – unless it has to cope with more than one people, so microcomputer systems *need* very little of this organisation.

With two or more people, the system has to start looking after each person's privileges, property, security, money, and other factors essential for productive computing over possibly long periods of time. Other system characteristics are less essential, but helpful in establishing an environment which supports a personal style of work – so it might be helpful to establish automatic access to certain file directories, and to set up particular screen formats, terminal characteristics, and so on. Many systems therefore provide means to define such properties with parameters which the system can remember from session to session.

HOW ARE PEOPLE HANDLED ?

From the system's point of view, people are just one more thing to be looked after, much like devices or files or processes – so you'd expect something like an array of people descriptors. This is *not* a standardised part of operating system practice; the information is usually there somewhere, but might be scattered thinly through the system. We shall call this (possibly notional) collection of information about people the *user database*.

The structure of the user database might be *flat*, with all people having much the same status, or *hierarchic*, with some people having certain rights over others. In other systems, people might be associated into (not necessarily hierarchic) *groups*. In practice, the basic structure is almost always flat, with any more complex relationships implemented by links between records as required. In a large system, there is also likely to be some supporting structure to help with finding the right record given the user name; a straight search through several thousand records can take an inconveniently long time even on a fast machine.

WHAT INFORMATION IS NEEDED ?

The system must carry whatever information it needs to do these various jobs. We can think of this as information about *what people have* (possessions, such as files, electronic mail, privileges in the system) and *what people do* (communicate with the system, use resources, execute programmes).

This information about people comes in two varieties. Some information lasts from session to session, and must therefore be stored permanently somewhere in the system. This is the material of the user database. Further information is required for someone actively using the system; this is temporary, and can be thought of either as information about the person running the computer session or about the session itself. We'll discuss it here anyway.

THE USER DATABASE.

Here are some examples of items commonly found in a system's user database. This is not intended as an exhaustive list, and not all are found in every system, though each has appeared in some system which we have known; this is a set of examples, to show the sort of information that might be useful. These data might or might not be kept together, but should be integrated.

As with other system tables which we shall encounter later, the items in the table all have fixed sizes; the intention is to provide a widely accessible record of data related to some entity, and the preservation of a standard structure is important to guarantee that the system will continue to work; it is an example of system consistency. Items of unpredictable size or of unpredictable number are represented by pointers, and stored elsewhere.

IDENTITY	usercode	Your primary name in the operating system – otherwise called login name, userid, ID, etc.
	personal details	Name, address, telephone number, etc. Useful for the manual side of the administration.
	password(s)	Secret(s) between you and the system, which prove that you are who you claim to be.
	who pays the bills	For the accounting system. One customer may have several accounts.
	hierarchy links	Identifies your superior, your underlings, etc. Useful for hierarchic security systems.
	classification	Secretary, director, spy … – where you fit into the organisation. Useful for setting privileges.
PRIVILEGES	capabilities	Permissions to use certain parts of the system, other people's files, etc.
	limits on spending	"Funny money" – limits your total use of the system, typically renewed at regular intervals.
	limits on resource use	An alternative to funny money : explicit limits on your use of individual resources.
	group membership	A means of defining groups, usually to give special security privileges.

POSSESSIONS	state of current account(s)	Real money. Used by the accounting systems, and to deny access if overspent.
	pointer to files	How to find your files in the system. (Usually the name of your base directory.)
	letterbox	A receptacle for electronic mail or other communications.
HISTORY	session records	How many sessions you have used, how much time spent, etc. – to keep track of your work.
	when last logged in	Mainly useful as a quick check that no one else is using your account.
PARAMETERS	search path	An ordered list of directories to search when looking for named files.
	login sequence	Things to do every time you log in to the system – usually as a login file.
	terminal definitions	How you want the screen presented, definitions of special keys, etc.

The system has to ensure that any necessary security measures are enforced. Different parts of the system need various forms of access to different parts of the information : the owner, other people, administrators, the whole system, the login programme, etc. all need some sort of access.

DURING A SESSION.

Here are some examples of information which pertains to a session, and which the system must maintain in order to exercise proper management. Some of these items are initially set from the userdata information, but the values may in some cases be changed during the session.

Current position	Which terminal is associated with your usercode – so that mail, messages, etc. can be directed to you.
File system positions	Typically your base directory and a current working directory.
Charges for the session	Funny or real money.
Current parameters	Meanings for control or function keys, screen layouts.
History	Instructions recently issued, previous system states – used to repeat or modify instructions, or roll back computations to checkpoints.

Much of this information concerns your personal preferences for interactive work, so in older systems the collection of session information was typically smaller.

LOGGING IN.

To see how some of these data are used, here's a description of a fairly typical logging-in sequence for a shared computer system.

1. Find the usercode in the user registration file.

   Someone has tried to log in, and has presented a usercode. Use the access structure to search the userdata file for the corresponding record. If it isn't there, don't tell anybody yet – if you do, it's much easier for criminals to identify good usercodes. Go through a dummy password sequence anyway.

2. Check the password.

   If it was a good usercode, check that the person knows the correct password. If not, or if the usercode was wrong, report the error. (Don't say what the error was – just that the sequence was wrong somewhere.) Otherwise, carry on.

3. Can this person use this terminal now ?

   Check the access rights and account balance from the userdata. Some systems are quite fussy – you may be able to use only certain terminals, or only certain hours during the day. Others are much more permissive. If access is denied at this point, say why; you know it's the right person now (you hope).

4. Set up the session control information from records.

   Find the basic parameters which determine some normal starting point for the session. These are properties which are defined for each usercode, but only those which don't significantly change the behaviour of the system – such as the base of the person's file directory, resource usage limits. We need the basic data, but we shall want to perform several further operations, so we don't want to install any sort of elaboration on the basic system behaviour yet. In some systems, this sort of information is held in an explicit structure, sometimes called a session control block.

5. Set initial values for this session.

   Set initial values for the parameters of this session. These might include the resources used in the session (initially zero), a channel number which identifies the terminal or workstation (in a shared system), the current working directory (initially the base directory).

6. Look for messages from other people.

   Look for any messages which might be waiting for this usercode. This is now usually electronic mail, but some systems might still send messages of other sorts about system conditions, changes, or events.

7. Record the login in the history.

   Record the login in the system log, and the usercode's history record, if any.

8. Set the appropriate session parameters.

   Change the configuration to suit the person's preferences, if any. We can afford to do this now, as all the important and sensitive tasks are complete. In this step screen layouts, search profiles, and other personal preferences may be set up, and perhaps programmes executed to perform desired initial operations.

9. Report successful login.

   Report successful login by some appropriate message and await instructions from the terminal.

Obviously enough, not all systems will perform just these tasks in just this sequence, but this list should make the point that the structure of a computer session is not trivially simple, and must be set up with some care.

COMPARE :

Lane and Mooney[INT3], Chapter 17, section 6.

QUESTIONS.

What are the implications of distributed systems for the problems of managing people ?
