ECOLE POLYTECHNIQUE

TRAVAUX DIRIGES SUR L'UTILISATION DU SYSTEME D'EXPLOITATION DES ORDINATEURS UNIX

P. COUSOT

EDITION 1987/88

TABLE DES MATIERES

(Les paragraphes marqués d'une étoile (*) sont à réserver pour une deuxième lecture. Les paragraphes marqués de deux étoiles (**) ou trois étoiles (***) pourront être consultés à la demande en cas de besoin pendant les projets de programmation).

TRAVAUX DIRIGES SUR L'UTILISATION DU SYSTEME D'EXPLOITATION DES ORDINATEURS UNIX

P. COUSOT

Un micro-ordinateur de type Macintosh a une puissance beaucoup trop limitée pour faire des calculs scientifiques importants. Nous utiliserons donc également un ordinateur Vax 8600 ou des stations de travail SPS 7/300, les Macintoshs servant alors de terminaux alpha-numériques et graphiques.

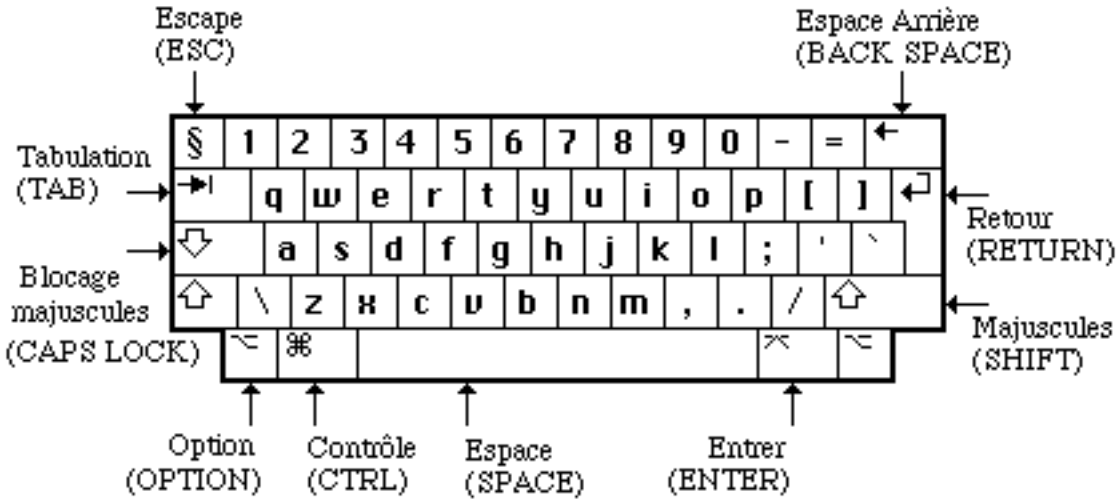
Un ordinateur peut aussi être utilisé pour organiser, conserver, mettre à jour, partager ou échanger des informations, c'est-à-dire à la fois comme un téléphone, un service postal, une bibliothèque, une imprimerie, etc... Pour cet usage, l'ordinateur dispose d'un grand nombre de logiciels de base dont l'ensemble s'appelle le *système d'exploitation*.

Cette séance de travaux dirigés est une introduction très élémentaire à l'utilisation du système d'exploitation Unix que nous avons retenu parce qu'il est à la fois simple, pratique et très répandu depuis les micro-ordinateurs jusqu'aux super-calculateurs.

Nous commençons par donner un exemple de session de travail qui introduit les commandes de première nécessité pour utiliser le Vax 8600 comme serveur de puissance sous PASCAL. Dans une deuxième partie nous introduisons des commandes supplémentaires, en particulier d'édition de texte, qui s'avèreront certainement vite très utiles, voire indispensables. Nous avons reporté en annexe des compléments qui pourront être étudiés au fur et à mesure des besoins.

1. CLAVIER

Nous rappellons ci-dessous les noms des touches spéciales du clavier du Macintosh. Nous donnons également le nom ou l'abréviation en Anglais (qui figure parfois sur les claviers informatiques classiques):



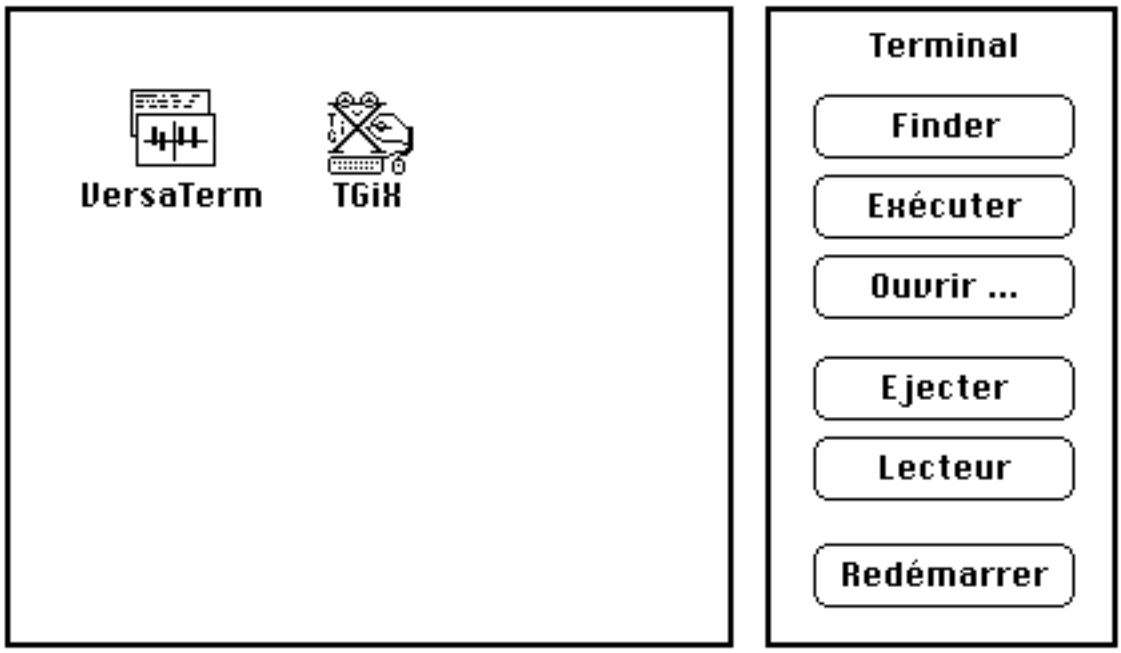
CTRL x (ou ^x dans la documentation UNIX) dénote le fait de maintenir la touche CTRL enfoncée puis de taper x.

DEL n'existe pas sur le Macintosh mais s'obtient en enfonçant OPTION puis en tapant ESC.

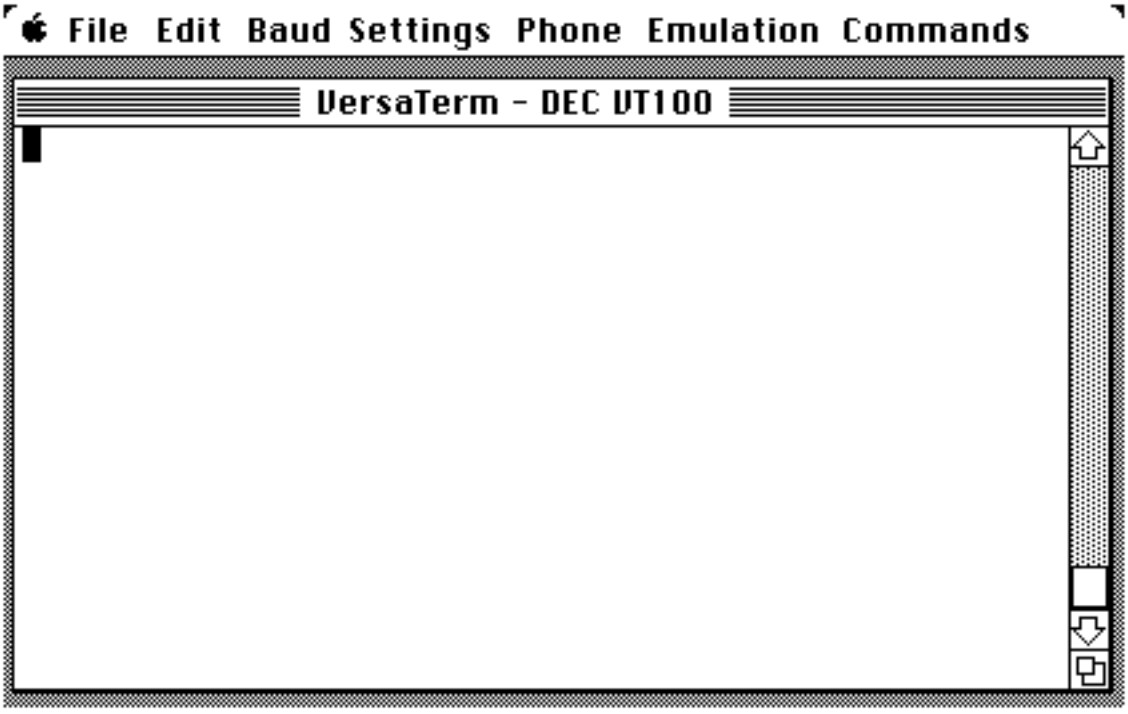
2. CONNEXION, CHANGEMENT DU MOT DE PASSE, DECONNEXION

2.1 CONNEXION AU VAX 8600

Mettez le Macintosh en marche et introduisez la disquette "Terminal" dans le lecteur interne. Il apparaît l'écran suivant:



Sélectionner **VersaTerm** puis **Exécuter**. Le Macintosh est maintenant un terminal du Vax 8600 (de type VT 100), dont l'écran a la forme suivante:



(Vérifier que la vitesse de transmission de `9600` bits par seconde est sélectionnée dans le menu `Baud`, que les options `Xon/Xoff`, `Parity:None`, `Char Size:8 bits` et `Stop Bits:1.0` sont sélectionnées dans le menu `Settings` et que le terminal `DEC VT100` est sélectionné dans le menu `Emulation`).

Après avoir tapé RETURN, le message :

```
Ultrix T1.2-2 (poly)
login:
```
apparaît sur l'écran. Tapez votre *nom de bord* en minuscules (les touches SHIFT et CAPS LOCK n'étant pas enfoncées) puis RETURN (Ce nom de bord est constitué par les 8 premières lettres de votre nom). La réponse est constituée par le texte:

```
Password:
```

Tapez ensuite votre *mot de passe* provisoire qui vous a été remis sous enveloppe cachetée. Le texte tapé n'apparaît plus sur l'écran pour que ce mot de passe reste secret.

En cas d'erreur, on est prévenu par le message :

```
Login incorrect
```

Il faut alors attendre l'apparition de `login:` pour recommencer.

Quand la connexion est établie, on obtient divers messages et enfin le *caractère d'incitation* `%` apparaît en début de ligne pour indiquer que le travail sur le Vax 8600 peut commencer.

2.2 CHANGEMENT DU MOT DE PASSE

La première opération à effectuer sur le Vax 8600 est de remplacer le mot de passe qui vous a été remis. Ce mot de passe provisoire a été tiré aléatoirement. Il peut être remplacé par un mot personnel plus facilement mémorisable. Votre mot de passe ne doit pas être trop simpliste et doit toujours rester secret. C'est le seul moyen d'éviter que des personnes extérieures à l'Ecole Polytechnique ayant accès au Vax 8600 par téléphone puissent se connecter.

Quand le caractère d'incitation `%` apparaît en début de ligne, vous pouvez changer votre mot de passe en tapant la commande:

```
passwd
```

Vous devez fournir l'ancien mot de passe puis le nouveau, deux fois de suite, pour éviter les erreurs. En cas d'oubli de ce nouveau mot de passe personnel, seul l'ingénieur système peut vous permettre de le changer.

2.3 DECONNEXION

Avant de quitter le Macintosh vous devez impérativement vous déconnecter en tapant:

```
logout
```

puis en sélectionnant l'option `Quit` du menu `File`. En cas d'oubli, un utilisateur du Macintosh que vous avez quitté pourrait avoir accès aux fichiers que vous conserverez sur le Vax 8600 (et ceci même si le Macintosh a été éteint).

3. UTILISATION MINIMALE DU VAX 8600

Dans ce paragraphe nous décrivons une utilisation minimale du Vax 8600 pour sauvegarder des fichiers, compiler et exécuter des programmes PASCAL, en présentant une session de travail type.

3.1 CREATION D'UN FICHIER SUR LE MACINTOSH

Après connexion au VAX 8600, utilisez l'éditeur de textes `MockWrite` pour créer le programme PASCAL suivant:

```
program Factorielle (input, output);
 var N, F : integer;
begin
 write('N?: '); readln(N);
 write(N : 1, '! = ');
 F := 1;
 while N <> 0 do begin
  F := N * F;
  N := N - 1;
 end;
 writeln(F : 1);
end.
```

Sauvegarder ce texte sur votre disquette "`Exercices`" dans un fichier de nom "`factorielle.p`".

3.2 INTERPRETEUR DE COMMANDES

Après connexion, le Vax 8600 va exécuter l'*interpréteur de commandes* (appelé `csh`) qui va indéfiniment :

- afficher le symbole d'incitation `%` en début de ligne,
- puis attendre que vous tapiez une commande au clavier (suivie d'un RETURN à ne pas oublier),
- puis exécuter la commande (ou indiquer qu'elle est incorrecte par un message d'erreur),
- puis recommencer (jusqu'à la fin de la session marquée par `logout`).

Voici quatre exemples de commandes que vous pouvez essayer:

`date`	imprime la date et l'heure,
`whoami`	imprime le nom de bord (à utiliser quand on trouve un terminal inoccupé),
`who`	donne la liste des noms de bord des utilisateurs se servant de l'ordinateur pour le moment,
`w`	idem, en décrivant les activités de chaque utilisateur.

3.3 FAUTES DE FRAPPE ET ERREURS

Quand on fait une faute de frappe dans une commande, on peut la corriger (avant d'enfoncer la touche RETURN) comme suit:

- la touche BACKSPACE permet d'effacer le dernier caractère tapé,
- la touche @ permet d'effacer toute la ligne.

Par exemple "`adte@dar`BACKSPACE`te`" équivaut à "`date`".

Les caractères @ et BACKSPACE peuvent être transmis tels quels (et non comme des caractères de correction) en les faisant précéder de \ (de sorte que pour effacer un \ il faut deux BACKSPACEs).

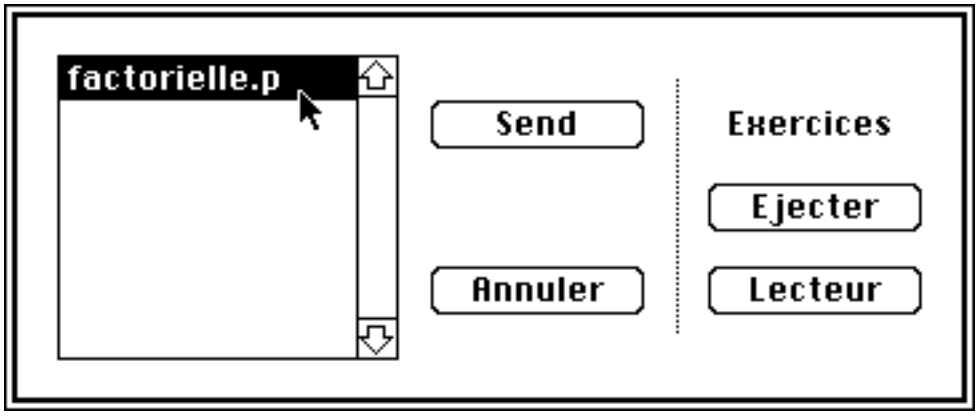
3.4 TRANSFERT D'UN FICHIER DEPUIS UNE DISQUETTE DU MACINTOSH SUR LE VAX 8600

Pour transférer le fichier `factorielle.p` depuis la disquette `Exercices` sur le VAX8600, suivre les instructions suivantes:

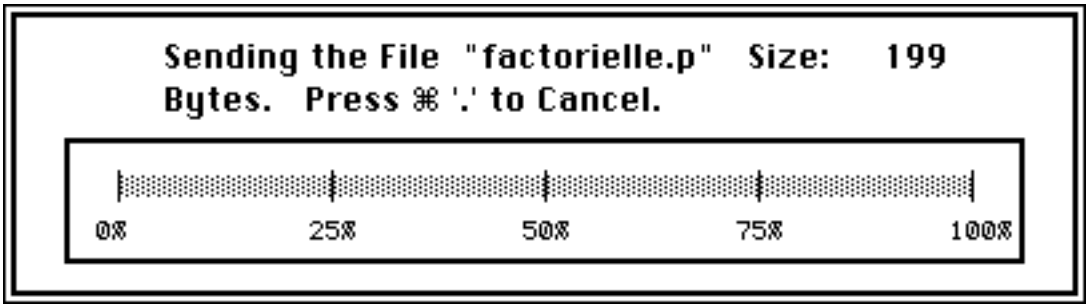
- positionner l'option `Kermit` dans le menu `File` (si elle n'y est pas déjà).
  Pour se faire, il suffit de sélectionner l'option `Mac XModem` du menu `File` qui est alors remplacée par l'option `Text XModem`. Quand cette option est sélectionnée, elle est remplacée par `MacBinary XModem`. En sélectionnant cette dernière option, on obtient `Kermit`.

- tapez la commande:
  `kermit -p n -r`

- sélectionner l'option `Send File...` du menu `File`. Il apparaît une fenêtre de dialogue qui permet de sélectionner le lecteur contenant la disquette `Exercices` puis le fichier `factorielle.p` et enfin `Send` (envoyer):



La fenêtre de dialogue suivante rend compte du déroulement du transfert:



Cette fenêtre disparaît après transmission du fichier sur le Vax 8600 qui écrit le caractère d'incitation `%` en début de ligne.

Si un fichier portant le nom du fichier transféré existe déjà sur le Vax 8600, ce fichier sera automatiquement détruit et remplacé par le fichier transféré. Ceci peut conduire à la perte de fichiers importants. Pour l'éviter, on peut utiliser:

`kermit -p n -w -r`

qui protège les fichiers du Vax 8600 en écriture ou la commande `ls` qui donne la liste de vos fichiers sur le Vax 8600 puis la commande:

```
kermit -p n -r -a nfichier
```

qui enregistre le fichier envoyé par le Macintosh sur le Vax 8600 sous le nom `nfichier` (et non plus sous le nom qu'il porte sur le Macintosh).

3.5 GESTION ELEMENTAIRE DES FICHIERS SUR LE VAX8600

3.5.1 LISTE DES FICHIERS

La commande (**l**i**s**t):

```
ls
```

donne la liste des noms des fichiers enregistrés sous votre nom de bord sur le Vax 8600. En tapant cette commande vous devez retrouver le nom du fichier "`factorielle.p`".

3.5.2 LISTAGE D'UN FICHIER

Pour lister un fichier sur l'écran, on utilise généralement la commande:

```
more nfichier
```

où `nfichier` est le nom du fichier à lister. Essayez par exemple:

```
more factorielle.p
```

Le listage a lieu par pages successives de la taille d'un écran. Pour lister un fichier qui fait plus d'un écran, la commande `more` accepte à son tour les commandes suivantes:

RETURN	pour lister une ligne supplémentaire,
SPACE	pour lister une page supplémentaire,
`q`	pour interrompre le listage et quitter la commande `more`,
`h`	pour obtenir un résumé des commandes de `more`.

3.5.3 IMPRESSION D'UN FICHIER

Pour imprimer un fichier de nom `nfichier` sur l'imprimante rapide en libre service, on utilise la commande (**l**ine **pr**inter):

```
lpr -p nfichier
```

La liste commence par le nom de bord et le nom du fichier en gros caractères puis le contenu du fichier est listé par pages de 66 lignes.

3.5.4 RECOPIE D'UN FICHIER

Pour recopier un fichier `nfichier1` dans un fichier `nfichier2`, on utilise la commande (**co**p**y**):

```
cp nfichier1 nfichier2
```

Si le fichier `nfichier2` n'existait pas, il est créé. On peut le vérifier par la commande `ls`:

```
% cp factorielle.p fact.p
% ls
fact.p        factorielle.p
```

Si le fichier existait déjà, le système demande à l'utilisateur si l'ancien fichier doit être détruit (`overwrite nfichier?`). La copie est effectuée si la réponse est `y` et n'est pas effectuée quand la réponse est différente (`n` par exemple).

3.5.5 CHANGEMENT DU NOM D'UN FICHIER

Pour changer le nom de `nfichier1` en `nfichier2`, on utilise la commande (**m**o**v**e avec l'option d'**i**nteraction):

```
mv -i nfichier1 nfichier2
```

Si un fichier de nom `nfichier2` existe déjà, une question est posée (`remove nfichier2?`) pour savoir si celui-ci doit être détruit (répondre `y`) ou bien s'il faut annuler la commande (répondre `n`).

3.5.6 SUPPRESSION D'UN FICHIER

Pour supprimer un fichier, on utilise la commande (re**m**ove):

```
rm nfichier
```

Avant de supprimer le fichier, le système demande à l'utilisateur de confirmer (`y`) ou de se rétracter (`n`):

```
% rm factorielle.p
rm:  remove factorielle.p? n
% ls
fact.p        factorielle.p
% rm factorielle.p
rm:  remove factorielle.p? y
% ls
fact.p
```

3.6 COMPILATION D'UN PROGRAMME PASCAL SUR LE VAX 8600

La *compilation* (c'est-à-dire la traduction en langage machine) d'un programme PASCAL (enregistré dans un fichier `nfichier.p` dont le nom doit impérativement se terminer par `.p`) se fait automatiquement à l'aide d'un programme appelé *compilateur* que l'on met en oeuvre par la commande (**p**ascal **c**ompiler):

```
pc -C nfichier.p
```

Si aucune erreur syntaxique n'est detectée, il n'y a aucun message et le *programme machine* est rangé dans le fichier **`a.out`** (qui est détruit s'il existait déjà):

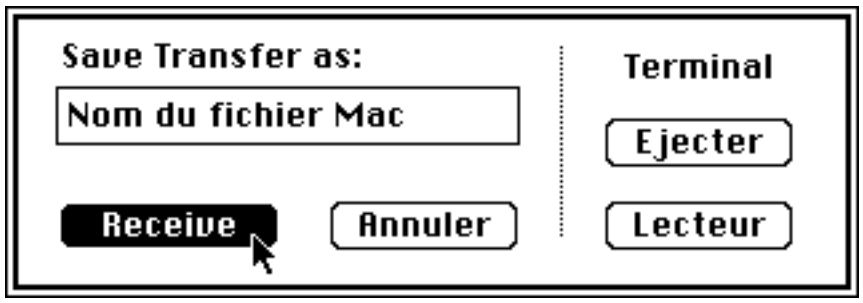
```
% pc -C fact.p
% ls
a.out         fact.p
```

En cas d'erreurs syntaxiques on peut utiliser un éditeur de texte (comme **`vi`** introduit plus loin) pour les corriger sur le Vax 8600. Il est également possible de rapatrier le fichier `nfichier.p` sur le Macintosh, de le corriger (par exemple avec MockWrite) puis de le renvoyer sur le Vax8600 (qui reste connecté pendant ce temps là).

3.7 TRANSFERT D'UN FICHIER DEPUIS LE VAX 8600 SUR UNE DISQUETTE DU MACINTOSH

Pour transférer le fichier de type texte `nfichier` depuis le Vax 8600 sur une disquette du Macintosh, suivre les instructions suivantes:

- positionner l'option `Kermit` dans le menu `File` (si elle n'y est pas déjà, voir § 3.4).
- tapez la commande:
  `kermit -p n -s nfichier`
- sélectionner l'option `Receive File...` du menu `File`. Il apparaît une fenêtre permettant de choisir la disquette et le nom sous lequel le fichier doit être enregistré sur le Macintosh:



(Si un fichier porte déjà ce nom, une seconde fenêtre de dialogue permet ou non de le remplacer). Il apparaît ensuite une autre fenêtre de dialogue pour rendre compte du déroulement du transfert:

Quand le transfert est terminé, cette fenêtre disparaît et le Vax 8600 écrit le caractère `%` en début de ligne.

3.8 EXECUTION D'UN PROGRAMME MACHINE

Pour exécuter un programme machine contenu dans un fichier `nfichier` du Vax 8600, il suffit de taper le nom **`nfichier`** de ce fichier. On obtient par exemple:

```
% a.out
N?: 3
3! = 6
% a.out
N?: 6
6! = 720
```

3.9 ARRET FORCE DE L'EXECUTION D'UNE COMMANDE UNIX OU D

Quand l'exécution d'un programme boucle indéfiniment ou bien quand une commande ne se déroule pas normalement, on peut les arrêter par CTRL `c`.

Si vous essayez, par exemple, de taper la commande:

```
yes
```

l'ordinateur va boucler indéfiniment en imprimant `y` sur l'écran. On peut l'arrêter par CTRL `c`. En général l'effet n'est pas immédiat car, par exemple, il faut attendre que tous les `y` envoyés par le Vax 8600 entre `yes` et CTRL `c` aient été écrits sur l'écran et il peut y en avoir un nombre considérable.

3.10 MANUEL DE REFERENCE EN LIGNE

Le *manuel du programmeur* UNIX décrit en détail toutes les commandes. Ce manuel est disponible en permanence sur l'ordinateur et peut être consulté au moyen de la commande `man`. Par exemple, la commande `cal` permet d'imprimer un calendrier. On peut en obtenir la description par :

```
man cal
```

ce qui donne :

```
                                                                CAL(1)

NAME
     cal - print calendar

SYNTAX
     cal [ month ] year

DESCRIPTION
     Cal prints a calendar for the specified year.  If a month is
     also  specified,  a calendar just for that month is printed.
     Year can be between 1 and  9999.   The  month  is  a  number
     between 1 and 12.  The calendar produced is that for England
     and her colonies.

     Try September 1752.

RESTRICTIONS
     The year is always  considered  to  start  in  January  even
     though this is historically naive.

                                                                  0-1
--More--(99%)
```

L'impression s'arrête quand l'écran est plein. Comme pour la commande `more`, il faut frapper la barre d'espacement pour obtenir une page d'écran supplémentaire. Dans notre exemple on n'obtient qu'une ligne blanche supplémentaire.

Pour décrire la syntaxe de la commande, les conventions suivantes sont utilisées:

- les crochets **`[]`** autour d'un argument de la commande indiquent que cet argument est optionnel,
- les trois points **`...`** indiquent qu'il peut y avoir un ou plusieurs arguments similaires.

Essayez, par exemple :

```
man whatis
man apropos
man man
```

Quand on recherche des informations sur une commande dont on ne connait pas le nom, la commande `man` est inutilisable. Pour ce faire, la commande `about` permet de rechercher toutes les commandes Unix en rapport avec un mot-clé (puis d'enchaîner avec une exécution de man pour la commande que l'on aura choisie).

Essayez, par exemple :

```
about pascal
```

3.11 FIN DE LA SESSION DE TRAVAIL

Pour sortir de la boucle de l'interpréteur de commandes, il faut taper la commande **`logout`** quand `%` apparait sur l'écran.

3.12 RESUME DES COMMANDES UNIX DE BASE

`about key`	listes des commandes en rapport avec le mot-clé `key`,
`a.out`	exécuter le programme machine `a.out`,
`cp nf1 nf2`	copier le fichier `nf1` dans le fichier `nf2`,
`kermit -p n -r`	recevoir un fichier (envoyé par `Send File...` du menu `File` (avec `Kermit`) sur le Macintosh) en conservant le même nom,
`kermit -p n -w -r`	idem, en protégeant les fichiers du Vax 8600,
`kermit -p n -r -a nf`	idem, en donnant le nom `nf` au fichier reçu,
`kermit -p n -s nf`	envoyer le fichier `nf` (reçu par `Receive File...` du menu `Fichier` (avec `Kermit`) sur le Macintosh),
`logout`	fin de la session de travail,
`lpr -p nf`	listage du fichier `nf` sur imprimante rapide,
`ll`	liste des noms des fichiers avec la date de dernière modification et la taille,
`ls`	liste des noms des fichiers,
`man cmd`	documentation en ligne sur la commande `cmd`,
`more nf`	listage du fichier `nf` par pages d'écran,
RETURN	ligne supplémentaire,
SPACE	page supplémentaire,
`q`	quitter `more`,
`h`	résumé des commandes de `more`,
`mv -i nf1 nf2`	changer le nom `nf1` du fichier en `nf2`,
`passwd`	changer le mot de passe,
`pc -C nf.p`	compiler le fichier PASCAL `nf.p`, résultat dans `a.out`,
`rm nf`	supprimer le fichier `nf`.
BACKSPACE	effacer le dernier caractère,
@	effacer la ligne,
CTRL `c`	arrêt de l'exécution .

4. EDITEUR DE TEXTES `vi`

4.1 INITIALISATION

Pour créer un fichier de texte portant le nom `nfichier`, on utilise la commande:

```
vi  nfichier
```

Il apparaît alors sur l'écran:

```
~
~
~
...
~
~
~
"nfichier" [New file]
```

ce qui indique qu'il s'agit d'un nouveau fichier appelé `nfichier` qui, pour l'instant, est vide.

Si le fichier existait déjà, les premières lignes du texte contenu dans le fichier apparaissent sur l'écran, le curseur étant positionné en haut à gauche de l'écran.

4.2 INSERTION DE TEXTE DANS UN FICHIER VIDE

Pour insérer du texte dans un fichier vide, on tape la commande (**i**nsert) **`i`** puis le texte (en corrigeant les fautes de frappe dans la ligne courante par BACKSPACE et @). Chaque fin de ligne est marquée par RETURN**.** L'insertion du texte se termine par ESC. On passe alors en mode correction.

4.3 CORRECTION DU TEXTE

On ne voit en permanence sur l'écran qu'une partie du texte (appelée fenêtre) et les corrections sont relatives au curseur:

La correction commence donc généralement par des déplacements de la fenêtre et du curseur (sans que la souris du Macintosh puisse être utilisée à cet effet) puis consiste à effacer et/ou insérer du texte.

4.3.1 Déplacement de la fenêtre

CTRL **`b`**	(**b**ackward) déplace la fenêtre en arrière (sauf si elle est au début du fichier),
CTRL **`f`**	(**f**orward) déplace la fenêtre en avant (sauf si elle est à la fin du fichier).

4.3.2 Déplacement du curseur

`0`	déplace le curseur au début de la ligne courante (il s'agit du chiffre zéro et non de la lettre majuscule O),
`$`	déplace le curseur à la fin de la ligne courante,
SPACE	déplace le curseur d'un caractère vers la droite (sauf s'il est en fin ligne),
BACKSPACE	déplace le curseur d'un caractère vers la gauche (sauf s'il est en début de ligne),
`-`	déplace le curseur sur la ligne précédente (sur le premier caractère non blanc (ou en fin de ligne) avec recul éventuel de la fenêtre d'une ligne quand le curseur est en bas de la fenêtre sans être au début du fichier),
RETURN	déplace le curseur sur la ligne suivante (sur le premier caractère non blanc (ou en fin de ligne) avec avance éventuelle de la fenêtre d'une ligne quand le curseur est en bas de la fenêtre sans être à la fin du fichier).

4.3.3 Effacement de texte

`x`	efface le caractère sous le curseur
`dd`	efface la ligne sous le curseur

4.3.4 Insertion de texte

`i`…ESC	(**i**nsert) insère le texte … à gauche du curseur,
`a`...ESC	(**a**ppend) insère le texte ... à droite du curseur,
`O`...ESC	(**O**pen) insère le texte ... sur une nouvelle ligne au-dessus du curseur (lettre O majuscule),
`o`...ESC	(**o**pen) insère le texte ... sur une nouvelle ligne en dessous du curseur (lettre o minuscule).

Pendant l'insertion du texte ... la touche BACKSPACE peut servir pour effacer le dernier caractère tapé et @ pour effacer tous les caractères de la ligne.

Les fins de lignes sont marquées par RETURN. En particulier, `i` RETURN ESC coupe la ligne courante en deux à gauche du curseur.

4.3.5 Erreurs de manipulation

Il faut noter que l'insertion de texte continue tant qu'elle n'est pas arrêtée par ESC. Quand il y a doute sur le mode de fonctionnement (en insertion ou correction), utiliser la touche ESC deux fois : l'éditeur passe alors en mode correction (s'il n'y était déjà) et indique par un bip qu'il est prêt à accepter une commande.

Les commandes d'édition prennent effet tout de suite sans s'imprimer sur l'écran. En cas d'erreur, un bip retentit. Si l'on se trompe de commande, on peut annuler son effet en tapant `u` (**u**ndo).

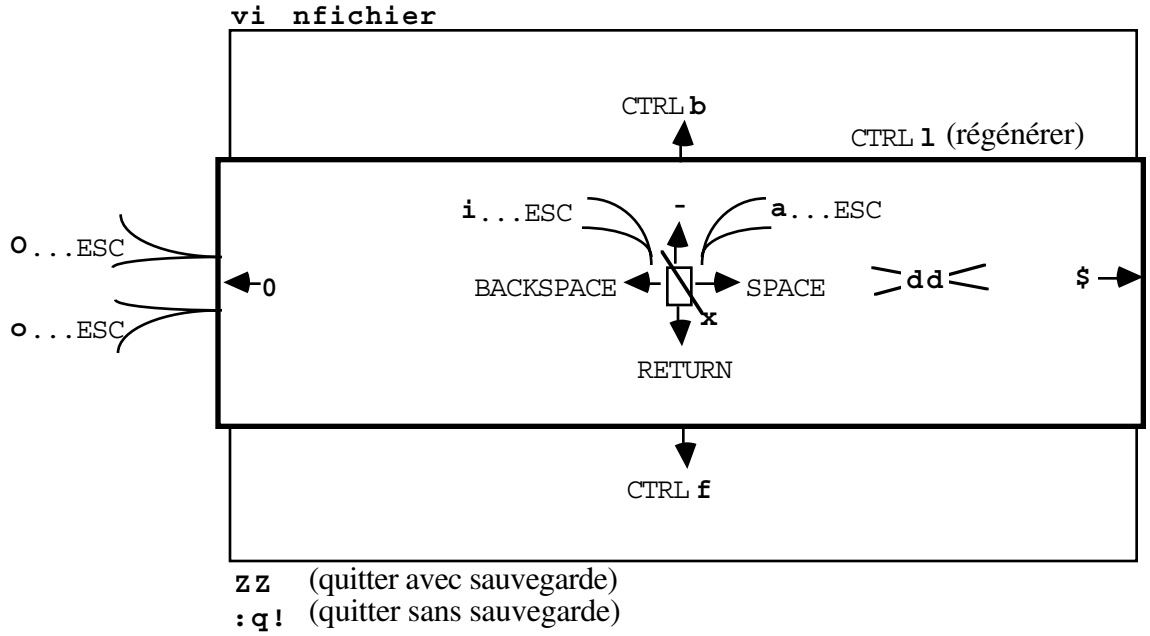
Quand l'écran est brouillé (par exemple à la réception d'un message de l'opérateur), taper CTRL `l` pour le regénérer (lettre L minuscule).

4.4 FIN DE L'EDITION DE TEXTE

Quand l'édition du texte est terminée, il faut taper **ZZ** (en majuscules) pour sauvegarder les modifications qui ont été apportées au fichier et revenir à l'interpréteur de commandes (qui imprime % en début de ligne). En tapant **:q!**, on quitte l'éditeur sans enregistrer le fichier (qui n'est donc pas créé ou pas modifié s'il existait déjà).

4.5 RESUME DES PRINCIPALES COMMANDES DE vi

Nous avons donné dans ce paragraphe un nombre minimum de commandes que l'on peut résumer par le schéma suivant :



ZZ (quitter avec sauvegarde)
:q! (quitter sans sauvegarde)

Pour un usage confortable de l'éditeur **vi,** consulter la liste des commandes complémentaires données en annexe §9.1.2.

5. MISE AU POINT ET EXECUTION D'UN PROGRAMME PASCAL SOUS UNIX

5.1 COMPILATEUR PASCAL

En cas d'erreur dans un programme PASCAL, un interpréteur (comme sur le Macintosh) offre beaucoup plus de facilités qu'un compilateur pour la mise au point du programme mais au prix d'un temps d'exécution plus long, souvent d'un facteur 10 pour le même ordinateur.

Par défaut le compilateur PASCAL d'Unix ne place aucun test à l'exécution dans le programme machine engendré. Ceci peut conduire à des résultats complétement erronés:

```
%pc fact.p
%a.out
N?: 13
13! = 1932053504
```

L'option `-C` (C majuscule) indique au compilateur qu'il faut insérer des tests de cohérence dans le programme machine engendré (au prix d'une augmentation du temps d'exécution de 10 à 15%) mais les messages d'erreur ne sont pas toujours très explicites:

```
%pc -C fact.p
%a.out
N?: 13
13! =
Integer overflow
Trace/BPT trap
```

De plus dans le cas d'un programme dont l'exécution ne se termine pas, on ne dispose d'aucune information pour se rendre compte que le programme boucle si celui-ci n'imprime aucun résultat à l'écran. C'est le cas du programme suivant (dont il faut arrêter l'exécution par CTRL `c`):

```
%more boucler.p

program Boucler (input, output);
begin
 while true do ;
end.

%pc -C boucler.p
%a.out
^C%
```

Enfin indiquons que si un fichier contient un programme machine (par convention, nous suffixerons alors son nom par `.x`), on peut exécuter ce programme en tapant simplement le nom du fichier.

Considérer par exemple :

```
%pc -C fact.p
%cp a.out fact.x
%rm a.out
rm: remove a.out? y
%fact.x
N?: 12
12! = 479001600
```

Pour savoir quels sont les fichiers contenant des programmes en langage machine on utilise la commande :

```
%ls -F
boucler.p       fact.p          fact.x*
```

leur nom apparaît alors suivi d'une étoile *.

5.2 REFERENCES CROISEES (**)

Un autre outil lié au compilateur PASCAL sous Unix est `pxref` qui permet d'obtenir des *références croisées* c'est-à-dire une indication pour chaque identificateur d'un programme de la liste des occurrences de cet identificateur repérées par un numéro de ligne:

```
%pxref fact.p
     1  program Factorielle (input,output);

     2   var N, F : integer;

     3  begin

     4   write('N?: '); readln(N);

     5   write(N : 1, '! = ');

     6   F := 1;

     7   while N <> 0 do begin

     8    F := N * F;

     9    N := N - 1;

    10   end;

    11   writeln(F : 1);

    12  end.
```

```
F                       2     6     8     8    11
Factorielle             1
N                       2     4     5     7     8     9     9
input                   1
integer                 2
output                  1
readln                  4
write                   4     5
writeln                11

        9 identifiers        20 occurrences
```

5.3 INTERPRETEUR PASCAL

Il existe un interpréteur PASCAL sous Unix:

```
%pix fact.p
Execution begins...
N?: 13
13! =
Integer overflow

Program error
Do you wish to enter the debugger? yes

Entering debugger ... type 'help' for help.

error at line 8
    8     F := N * F;
> quit
%
```

ce qui permet de déterminer que l'opération incohérente dans le calcul de 13! est la multiplication à la ligne 8 du programme. On peut obtenir une liste du programme avec numérotation des lignes par la commande:

`cat -n nfichier`

Par exemple:

```
%cat -n fact.p
     1  program Factorielle (input,output);
     2   var N, F : integer;
     3  begin
     4   write('N?: '); readln(N);
     5   write(N : 1, '! = ');
     6   F := 1;
     7   while N <> 0 do begin
     8    F := N * F;
     9    N := N - 1;
    10   end;
    11   writeln(F : 1);
    12  end.
```

Si le fichier est très long, le défilement peut être trop rapide pour que la lecture soit possible. On peut arrêter le défilement par CTRL `s` et le reprendre par CTRL `q` ou bien encore utiliser:

`cat -n nfichier | more`

Essayez également:

```
%pix boucler.p
Execution begins...

Statement count limit of 500000 exceeded

Program error
Do you wish to enter the debugger? no

        Error in "Boucler"+1 near line 3.
Execution terminated abnormally.

500000 statements executed in 0007.100 seconds cpu time.
```

L'exécution s'arrête après avoir exécuté 500 000 instructions PASCAL (ce qui n'est qu'une présomption de non-terminaison).

On peut éviter d'avoir une limite sur le nombre d'instructions exécutées en utilisant:

`pix -p boucler.p`

Il faut alors arrêter l'exécution avec CTRL `c`.

Les commandes de mise au point de l'interpréteur PASCAL sont les suivantes:

`run`	commencer l'exécution du programme,
`cont`	continuer l'exécution,
`step`	exécuter un pas du programme,
`next`	exécuter un pas du programme en sautant les appels de procédures,
`trace` l	tracer l'exécution de la ligne l,
`trace` p	tracer les appels de la procédure p,
`trace` v	tracer les modifications de la variable v,
`trace` e **`at`** l	imprimer la valeur de l'expression e quand l'exécution passe à la ligne l,
`stop at` l	suspendre l'exécution à la ligne l,

stop in p	suspendre l'exécution quand la procédure p est appelée,
status	imprimer la liste des `trace` et `stop` en cours,
delete n	supprimer la `trace` ou le `stop` de numéro n,
call p	appeler la procédure p,
where	écrire la liste des procédures en cours d'appel,
print e	écrire la valeur de l'expression e,
whatis i	écrire la déclaration de l'identificateur i,
list l1,l2	lister les lignes l1 à l2 du programme,
edit nfichier	éditer le fichier `nfichier` (avec `vi`),
quit	quitter l'interpréteur PASCAL,
help	rappeler la liste des commandes ci-dessus.

Pour passer en mode mise au point dès le début de l'exécution du programme `nprog.p` faire:

```
pi  nprog.p
pdx
```

(la commande `pi nprog.p` crée un fichier `obj` que l'on peut également exécuter sans aide à la mise au point (sauf en cas d'erreur) en tapant **px**)

5.4 PROFILS D'EXECUTION (**)

Pour la mise au point d'un programme PASCAL il est parfois utile de disposer d'un *profil d'exécution* c'est-à-dire du nombre de fois que chaque instruction a été exécutée. On peut repérer ainsi les boucles qui ne se terminent pas ou les instructions jamais exécutées. Pour cela, il faut lancer l'interpréteur avec l'option `-z` :

```
%pix -z fact.p
Execution begins...
N?: 12
12! = 479001600
Execution terminated.

42 statements executed in 0000.00 seconds cpu time.
```

L'exécution a créé un fichier (`pmon.out`) contenant le profil d'exécution qui s'exploite comme suit:

```
%pxp -z fact.p
Berkeley Pascal PXP -- Version 2.12 (5/11/83)

Mon Sep 15 10:16 1986  fact.p

Profiled Mon Sep 15 10:16 1986

     1        1.---|program Factorielle(input, output);
     2             |var
     2             |    N, F: integer;
     3             |begin
     4             |    write('N?: ');
     4             |    readln(N);
     5             |    write(N: 1, '! = ');
     6             |    F := 1;
     7             |    while N <> 0 do begin
     8           12.---|    F := N * F;
     9                 |    N := N - 1
     9             |    end;
    11             |    writeln(F: 1)
    11             |end. { Factorielle }
```

En face d'une barre verticale, on a le nombre de fois où l'instruction repérée par cette barre a été exécutée.

5.5 REDIRECTION DES ENTREES-SORTIES

Quand un programme (et plus généralement une commande Unix) affiche des résultats, ils apparaissent sur la sortie standard, c'est-à-dire sur l'écran.

Il est possible de changer la sortie standard (`standard output`) d'une commande quelconque :

`cmd`

et de rediriger la sortie vers le fichier `nfichier` (nouvellement créé) par

`cmd > nfichier`

(on obtient le message `nfichier: File exists`) si ce fichier existait déjà).

Considérer par exemple :

```
%ls
boucler.p       fact.p          fact.x          fact1.p
%cat -n fact1.p
     1  program Factorielle (input,output);
     2   var N, F : integer;
     3  begin
     4   readln(N);
     5   F := 1;
     6   while N <> 0 do begin
     7    F := N * F;
     8    N := N - 1;
     9   end;
    10   writeln(F : 1);
    11  end.
%pc -C fact1.p
%a.out > res
3
%ls
a.out           fact.p          fact1.p
boucler.p       fact.x          res
%more res
6
```

Pour ajouter les résultats à la fin du fichier `nfichier` (qui doit déjà exister) on utilise la notation

`cmd >> nfichier`

(on obtient le message `nfichier: No such file or directory` s'il n'existe pas), ce qui donne, par exemple :

```
%a.out >> res
4
%more res
6
24
```

Sachant que **`cmd1 ; cmd2`** consiste à éxécuter la commande `cmd1` puis la commande `cmd2`, on obtient par exemple :

```
%a.out >> res ; a.out >> res
5
6
%more res
6
24
120
720
```

De la même façon pour les programmes ou les commandes Unix qui lisent des données sur l'entrée standard, il est possible de remplacer le clavier par un fichier de nom `nfichier` en écrivant:

`cmd < nfichier`

ce qui donne par exemple:

```
%a.out < res
720
```

Les deux possibilités peuvent évidemment se combiner, comme dans :

```
% a.out < res >> res
%more res
6
24
120
720
720
```

Pour calculer (3!)!, on peut utiliser les commandes :

```
%a.out > temp
3
%a.out < temp
720
%rm temp
rm: remove temp? y
```

où `temp` est un fichier temporaire détruit après utilisation. De manière équivalente, on peut écrire :

```
%a.out | a.out
3
720
```

et plus généralement **`cmd1`** | **`cmd2`** consiste à exécuter `cmd1` puis `cmd2` en utilisant la sortie de la première commande comme entrée de la seconde. Cette possibilité se combine avec les précédentes ci-dessous (la commande **`cat`** redirige l'entrée standard (la fin de fichier étant représentée par CTRL `d`) sur la sortie standard):

```
%cat > 3
3
%more 3
3
%a.out < 3 | a.out > 6!
%ls
3               a.out           fact.p          fact1.p
6!              boucler.p       fact.x          res
%more 6!
720
```

Les messages d'erreur (à la compilation ou à l'exécution) sont écrits dans le fichier standard de diagnostics (`standard error`) qui, par défaut, est l'écran. On peut rediriger la sortie des diagnostics de la commandes `cmd` vers le fichier `nfichier` (nouvellement créé) par:

```
cmd >& nfichier
```

5.6 GRAPHISME EN PASCAL SOUS TGiX (par Philippe Chassignet, (*))

5.6.1 Compatibilité des programmes Pascal sur le Macintosh et sous Unix (*)

Le Macintosh Pascal dispose de facilités de mise au point qui n'ont pas leur équivalent dans l'environnement standard Unix mais son intérêt est limité par les faibles capacités du Macintosh auxquelles s'ajoute la lenteur d'un programme interprété comparée à celle d'un programme compilé. Par contre, en utilisant le compilateur `pc` sur le VAX 8600, on bénéficie de la rapidité et des avantages (bibliothèques, mémoire virtuelle, etc...) d'un système puissant. Pour des calculs importants, une bonne méthode consiste donc à mettre son programme au point sur le Macintosh en utilisant des jeux d'essai puis à le transférer sur le VAX 8600 ou sur les SPS 7/300 pour l'exécuter sur les données réelles.

Malheureusement, le compilateur `pc` ignore les facilités offertes par le Macintosh Pascal, notamment pour la manipulation des chaînes de caractères, pour le graphique et pour l'utilisation de la souris.

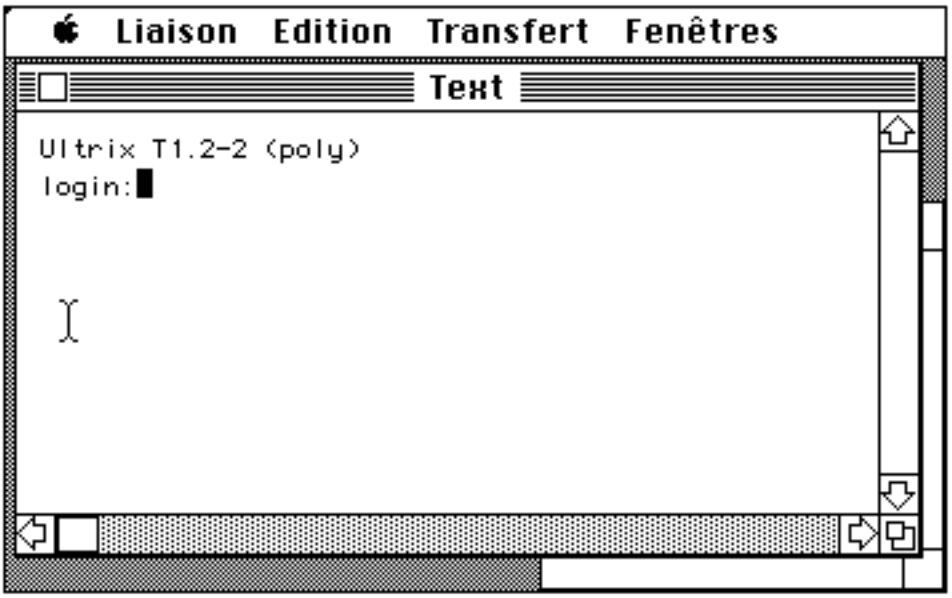
Le logiciel TGiX permet d'utiliser ces facilités sans renoncer à la compatibilité. En l'utilisant, un programme écrit sous Macintosh Pascal et respectant un minimum de contraintes peut être compilé et exécuté sous Unix après un minimum de transformations. La réciproque est également vraie.

5.6.2 Utilisation de TGiX comme terminal alpha-numérique (*)

5.6.2.1 Emulation d'un terminal VT100 (*)

TGiX peut être utilisé pour émuler un terminal alpha-numérique de type VT100. Pour lancer TGiX cliquer deux fois sur son icône:

On obtient l'écran suivant (réduit ci-dessous):



La fenêtre `Text` joue le double rôle de fenêtre terminal et de fenêtre d'entrées-sorties alphanumériques en Pascal. Elle cache la fenêtre `Drawing` servant de sortie graphique en Pascal.

L'option `Réinitialiser` du menu `Liaison` efface le contenu des fenêtres `Text` et `Drawing` et envoie le code CTRL c à Unix, ce qui a généralement pour effet d'arrêter le programme en cours d'exécution sous Unix.

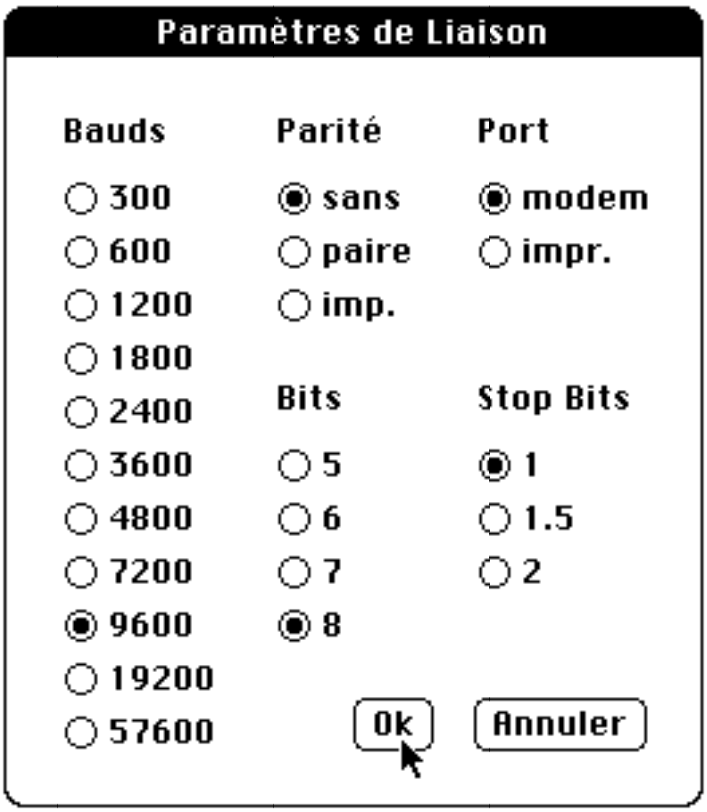
L'option `Quitter` du menu `Liaison` permet de quitter TGiX. On obtient la fenêtre suivante:

qui permet, en cas d'oubli, d'envoyer à Unix la commande de déconnexion `logout` avant de quitter TGiX.

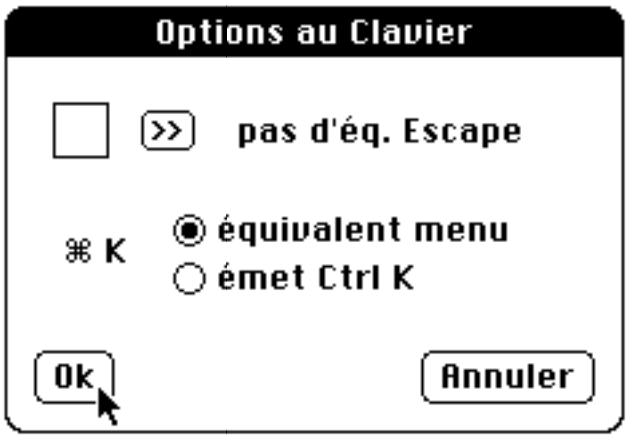
Le menu `Edition` a les options habituelles `Couper`, `Copier`, `Coller` qui ne sont utilisables que dans la fenêtre `Text` et sur une seule ligne à la fois.

5.6.2.2 Réglages des paramètres de communication de TGiX (***)

Les paramètres de communication sont réglés de manière standard pour le VAX 8600 (Bauds : 9600, Parité : sans, Bits : 8, Port : modem, Stop Bits : 1) Si nécessaire, l'option `Paramètres...` du menu `Liaison` fait apparaître la fenêtre ci-dessous qui permet de les régler différemment en cliquant sur les boutons correspondant à chaque paramètre :



De même les options d'utilisation du clavier sont réglées de manière standard mais l'option `Options...` du menu `Liaison` permet d'ouvrir la fenêtre suivante pour ajuster les réglages selon le type de Macintosh utilisé :



- Touche ESC :

Sur un Macintosh avec un clavier comportant une touche d'échappement ESC (marquée *esc* sur le Macintosh SE avec clavier étendu ou sur le Macintosh II) cliquer sur le bouton '**>>**' car la touche ESC existe déjà et permet d'envoyer le code correspondant à Unix.

Sur un Macintosh avec un clavier sans touche ESC (comme le Macintosh 512K), on peut définir une touche du clavier (généralement § car ce caratère n'est pas souvent utilisé sous Unix) qui permettra d'envoyer à Unix le code correspondant à ESC. Pour ce faire taper § puis cliquer sur le bouton '**>>**' pour obtenir:

- Touche CTRL :

Sur un Macintosh avec un clavier comportant une touche de contrôle CTRL (marquée *ctrl* sur Macintosh SE avec clavier étendu ou sur le Macintosh II) cliquer sur le bouton `équivalent menu` car la touche CTRL existe déjà et permet, en combinaison avec une autre touche d'envoyer les codes correspondant à Unix comme par exemple CTRL c. La touche ⌘ maintenue enfoncée peut donc être utilisée en combinaison avec une autre touche comme équivalent clavier des options du menu de TGiX. Par exemple ⌘ `Q` permet de quitter TGiX. Une frappe ⌘ `X` ne correspondant à aucun équivalent clavier restera sans effet.

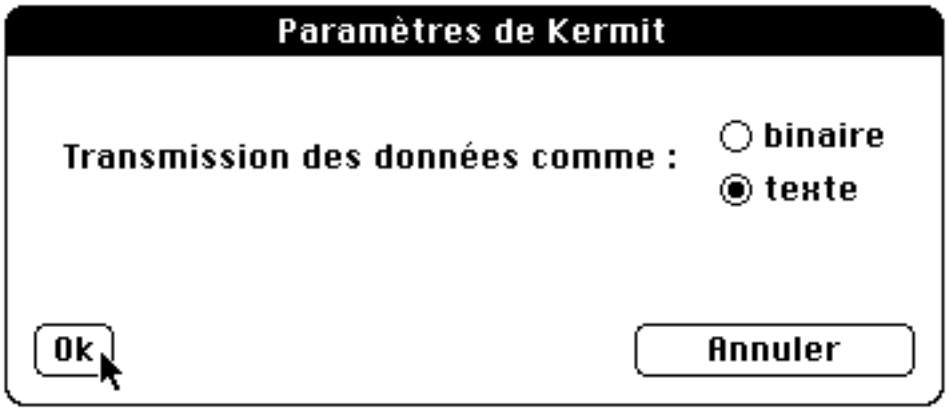
Sur un Macintosh avec un clavier sans touche CTRL (Macintosh 512K), on peut utiliser la touche ⌘ pour faire office de touche 'contrôle'. Pour ce faire cliquer sur le bouton `émet Ctrl K`. Dans ces conditions il n'est évidemment plus possible de sélectionner les options des menu de TGiX à l'aide de cette touche.Toute frappe ⌘ X est donc envoyée sous la forme du code correspondant à CTRL X à Unix.

5.6.3 Utilisation de TGiX pour le transfert de fichiers (**)

TGiX permet de transférer des fichiers de différent types entre le Macintosh et Unix en utilisant le logiciel de communication `kermit`.

5.6.3.1 Fichiers de type texte (**)

Pour transférer un fichier texte (par exemple un programme Macintosh Pascal, un fichier crée par l'éditeur de texte `Mockwrite` ou encore un fichier crée par l'éditeur de texte `MacWrite` et sauvegardé avec l'option `Text Only`) choisir l'option `Options...` du menu `Transfert` pour ouvrir la fenêtre ci-dessous dans laquelle on clique sur le bouton `Transmission des données comme texte` puis sur `OK`:



Lancer ensuite le transfert de fichier comme suit:

- Pour envoyer un fichier binaire de nom `nfichier` sous Unix vers le Macintosh:

. taper dans la fenêtre `Text` sous Unix la commande:

```
kermit -p n -i -s nfichier
```

. puis sélectionner l'option `Recevoir un texte` du menu `Transfert`.

- Pour envoyer un fichier texte de nom `nfichier` du Macintosh vers Unix:
  . taper dans la fenêtre `Text` sous Unix la commande:

```
kermit -p n -i -r
```

  . puis sélectionner l'option `Envoyer comme texte` du menu `Transfert`.

Dans les deux cas des fenêtres de dialogue permettent de choisir la disquette et le fichier concerné sur le Macintosh et de rendre compte du déroulement du transfert (avec la possibilité d'annuler le transfert en cas d'erreur de manipulation).

5.6.3.2 Fichiers binaires (**)

Les fichiers binaires (par exemple un fichier de type **`file of`** `integer` créé par un programme Macintosh Pascal) sont des fichiers dont le contenu dépend de la machine à laquelle ils correspondent et qui sont donc en général utilisables uniquement sur des machines du même type que celle sur laquelle ils ont été créés. Par exemple les objets Pascal de type **`file of`** `integer` n'ont pas la même représentation binaire sur le VAX 8600 et sur le Macintosh car la représentation des objets de type `integer` est différente sur ces deux machines. Les transferts de fichiers binaires entre machines différentes ne sont donc généralement utilisés qu'à des fins d'archivage.

Pour transférer un fichier binaire choisir l'option `Options...` du menu `Transfert` pour ouvrir une fenêtre dans laquelle on clique sur le bouton `Transmission des données comme binaire` puis sur `OK` puis lancer le transfert de fichier comme suit:

- Pour envoyer un fichier binaire de nom `nfichier` sous Unix vers le Macintosh:
  . taper dans la fenêtre `Text` sous Unix la commande:

```
kermit -p n -i -s nfichier
```

  . puis sélectionner l'option `Recevoir un binaire` du menu `Transfert`.
- Pour envoyer un fichier binaire de nom `nfichier` du Macintosh vers Unix:
  . taper dans la fenêtre `Text` sous Unix la commande:

```
kermit -p n -i -r
```

  . puis sélectionner l'option `Envoyer comme binaire` du menu `Transfert`.

Dans les deux cas des fenêtres de dialogue permettent de choisir la disquette et le fichier concerné sur le Macintosh et de rendre compte du déroulement du transfert (avec la possibilité d'annuler le transfert en cas d'erreur de manipulation).

5.6.3.3 Fichiers Macintosh (***)

Les fichiers du Macintosh (par exemple l'application `MacWrite` ou un fichier crée par cette application) ont une structure particulière qui ne permet pas de les transférer sous Unix en mode texte ou binaire et de les récupérer ensuite sous une forme utilisable par le Macintosh. Pour ce faire, on procédera comme suit:

- Pour envoyer un fichier Macintosh de nom `nfichier` sous Unix vers le Macintosh:
  . taper dans la fenêtre `Text` sous Unix la commande:

```
kermit -p n -i -s nfichier
```

  . puis sélectionner l'option `Recevoir une archive` du menu `Transfert`.
- Pour envoyer un fichier Macintosh de nom `nfichier` du Macintosh vers Unix:
  . taper dans la fenêtre `Text` sous Unix la commande:

```
kermit -p n -i -r
```

  . puis sélectionner l'option `Envoyer comme archive` du menu `Transfert`.

Dans les deux cas des fenêtres de dialogue permettent de choisir la disquette et le fichier concerné sur le Macintosh et de rendre compte du déroulement du transfert (avec la possibilité d'annuler le transfert en cas d'erreur de manipulation).

Le transfert de fichiers Macintosh sous Unix n'est utile qu'à des fins d'archivage et de distribution de logiciels.

5.6.4 Utilisation de TGiX comme terminal graphique sous Pascal (***)

5.6.4.1 Compilation et exécution d'un programme Pascal graphique sous Unix (***)

Soit à utiliser la version ***i*** de TGiX, supposée actuellement disponible.

Pour qu'un programme Macintosh Pascal utilisant des éléments du langage Pascal en dehors de la norme standard soit compilable sous Unix, il faut inclure la ligne suivante :

```
#include "/usr/local/graphic/include/MacLibi.h"
```

juste après l'en-tête du programme **`program ...`**. Ceci permet d'inclure au début du programme les éléments de Macintosh Pascal hors de la norme Pascal qui ont été redéfinis dans une bibliothèque Unix.

La compilation du programme ainsi modifié s'obtient par la commande :

```
gpc nomprog.p i
```

qui crée l'exécutable **`a.out`** dans le répertoire courant. (La commande `gpc` est un shell-script situé dans `/usr/local/bin` et dont vous pouvez faire une copie si vous désirez la personnaliser).

Lors de l'exécution des sous-programmes graphiques, le VAX envoie des séquences de caractères spécifiques au Macintosh, sur lequel doit donc tourner un programme, en l'occurrence `TGiX.i`, qui interprète ces séquences pour dessiner dans une fenêtre `Drawing`. Les séquences classiques émises par le shell, par un utilitaire Unix ou lors de l'exécution d'un `write` ou `writeln` dans votre programme sont interprétées différemment et affichées dans une fenêtre `Text`.

En cours d'exécution le menu `Fenêtres` permet d'ouvrir et de faire passer au premier plan l'une des fenêtres `Text` ou `Drawing` (les parties de la fenêtre `Drawing` ayant été cachées étant perdues comme sous Macintosh Pascal). L'option `Plein écran` du menu `Fenêtres` permet d'ouvrir la fenêtre active (dont le nom est repéré par une marque √ dans ce menu) au maximum des possibilités offertes par la taille de l'écran. L'option `Standard` permet d'ouvrir les fenêtres comme elles le sont de manière standard en Macintosh Pascal.

5.6.4.2 Performances et limites de TGiX (***)

La vitesse d'exécution des programmes graphiques est limitée par le débit des caractères émis vers le Macintosh. On n'obtient donc un gain de temps effectif, par rapport à une exécution sous Macintosh Pascal, ou même, Turbo Pascal que si le programme réalise un minimum de calculs entre chaque appel graphique.

Les temps de tracé (en secondes) observés sur quelques exemples du cours, sont significatifs, les programmes sont classés selon un rapport calcul/graphique croissant :

nom du programme	VAX + TGiX	Turbo Pascal	Macintosh Pascal
Sur Mac 512 :			
DessinerDragon (ordre 11)	35	3	57
DessinRecursifdunArbre (x40)	46	34	78
MethodeRungeKutta (1000 pas)	18	80	330
SurfacedeBoy	60	320	500
SeriedeFourier	12	680	???

5.6.4.3 Possibilités offertes par la version 6 de TGiX (***)

Les déclarations associées à cette version sont listées ci-dessous :

```
const
  Maxint = 32767;  { Borne supérieure des entiers du Macintosh }
  Maxlongint = maxint;  {     Borne supérieure des grands entiers du
                                                          Macintosh
  }

  srcCopy = 0;  { Les 16 modes de tracé du crayon }
  srcOr = 1;
  srcXor = 2;
  srcBic = 3;
  notSrcCopy = 4;
  notSrcOr = 5;
  notSrcXor = 6;
  notSrcBic = 7;
  patCopy = 8;
  patOr = 9;
  patXor = 10;
  patBic = 11;
  notPatCopy = 12;
  notPatOr = 13;
  notPatXor = 14;
  notPatBic = 15;

type
  Byte = 0 .. 255;  { Caractères du Macintosh }
  Integer = -32768 .. Maxint;  { Entiers du Macintosh }
  LongInt = integer;  { Grands entiers du Macintosh }
  Str255 = packed array[ 1..255 ] of char;
  Pattern = packed array[ 0..7 ] of Byte;  { Motifs de tracé du

                                                  crayon }
  Bits16 = array[ 0..15 ] of Integer;
  VHSelect =  ( v, h );
  S  t  y  l  e  I  t  e  m                                    =
  (bold,italic,underline,outline,shadow,condense,extend);
  Style = set of StyleItem;
```

```
      FontInfo = record                                    { En pixels :
      }
                              ascent : Integer;      { Ligne de base
      au haut du caractère }
                              descent: Integer;      { Ligne de base
      au bas du caractère  }
                                        widMax : Integer;      {
Largeur maximale d'un caractère    }
                              leading: Integer;      {   Distance
      entre lignes             }
                           end;

      Point = record case Integer of
                           0 : (      v : Integer;
                           h : Integer );
                           1 : (      vh : array[ VHSelect ] of
      Integer )
                     end;

      Rect = record case Integer of
                           0 : (      top : Integer;
                                      left : Integer;
                                      bottom : Integer;
                                      right : Integer ) ;
                           1 : ( topLeft : Point;
                                      botRight : Point )
                     end;

      Cursor = record
                     data : Bits16;
                     mask : Bits16;
                     hotSpot : Point;
               end;

      PenState = record
                     pnLoc : Point;
                     pnSize : Point;
                     pnMode : Integer;
                     pnPat : Pattern;
               end;

    var
      white : Pattern;  { Motifs prédéfinis de tracé du crayon }
      black : Pattern;
      gray : Pattern;
      ltGray : Pattern;
      dkGray : Pattern;
      arrow : Cursor;  { Curseurs prédéfinis }
      cross : Cursor;
      watch : Cursor;
      randSeed : LongInt;   { Pour générer des nombres pseudo-
      aléatoires }

    { Routines décrites dans le Macintosh Pascal Reference Manual [1] }

    { Manipulation de chaînes, voir leçon 2 }

      function length( str : Str255 ) : LongInt;
      function pos( substr, str : Str255 ) : LongInt;
```

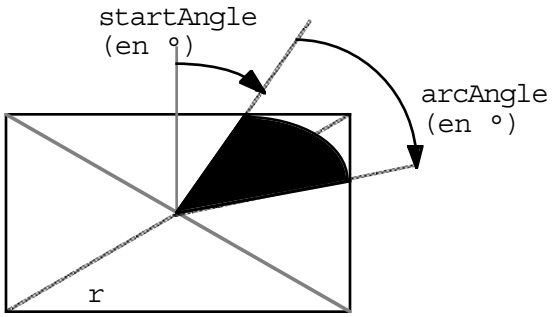
```
  function concat( str1, str2 : Str255 ) : Str255;
  function copy( source : Str255; index, count : Integer ) :
  Str255;
  procedure delete( var dest : Str255; index, count : Integer );
  function omit( str : Str255; index, count : Integer ) : Str255;
  procedure insert( source:Str255; var dest:Str255; index:Integer
  );
  function include( source,str : Str255; index : Integer) : Str255;

{ Gestion de la souris }

  function Button : boolean;
       {     Vrai ssi le bouton de la souris est enfoncé }
  function StillDown : boolean;
       {     Vrai ssi le bouton de la souris est resté enfoncé depuis
  la
             dernière action effectuée sur la souris }
  function WaitMouseUp : boolean;
       {     Pour tester un double-clique }
  procedure GetMouse( var x, y : Integer );
       {     Coordonnées locales du curseur déplacé par la souris }

{ Gestion des fenêtres Text et Drawing }

  procedure HideAll;
       { Cacher toutes les fenêtres }
  procedure ShowText;
       { Montrer la fenêtre 'Text' au premier plan }
  procedure ShowDrawing;
       { Montrer la fenêtre 'Drawing' au premier plan }
  procedure SetTextRect( WindowRect : Rect );
       { Définir les dimensions de la fenêtre 'Text' sur l'écran }
  procedure GetTextRect( var WindowRect : Rect );
       { Retourne les dimensions de la fenêtre 'Text' }
  procedure SetDrawingRect( WindowRect : Rect );
       { Définir les dimensions de la fenêtre 'Drawing' sur l'écran
  }
  procedure GetDrawingRect( var WindowRect : Rect );
       { Retourne les dimensions de la fenêtre 'Drawing' }

{ Temps }

  function TickCount : LongInt;
       {     Retourne le nombre de soixantièmes de secondes écoulés
  depuis
             le démarrage du Macintosh }


{ Routines décrites dans le Macintosh Pascal Technical Appendix [2]
  }

{ Extra QuickDraw }

  procedure DrawLine( a, b, c, d : Integer );
       { Trace un segment de (a,b) à (c,d) }
  procedure PaintCircle( x, y, r : Integer );
       { Trace un disque de centre (x,y) et rayon r }
  procedure InvertCircle( x, y, r : Integer );
```

```
        { Passe le disque de centre (x,y) et rayon r en vidéo-inverse
    }

{ Niveau GrafPort }

    procedure SetOrigin( h, v : Integer );
        { Origine du système de coordonnées locales du grafPort
    courant }
    procedure ClipRect( r : Rect );
        { Définit la zone de dessin du grafPort courant }
    procedure BackPat( pat : Pattern );
        { Définit le motif d'arrière-plan du grafPort courant }

{ Gestion du curseur }

    procedure InitCursor;
        {   Initialise le curseur. Sa forme est une flèche et son
    niveau
            est zéro. Le curseur suit automatiquement la souris }
    procedure SetCursor( crsr : Cursor );
        { Pour modifier la forme du curseur }
    procedure HideCursor;
        {   Cache le curseur et décrémente son niveau }
    procedure ShowCursor;
        {   Incrémente le niveau du curseur (si <0) et le fait
    apparaitre
            si son niveau est 0 }
    procedure ObscureCursor;
        {   Cache le curseur jusqu'au prochain déplacement de la
    souris }

{ Gestion du crayon }

    procedure HidePen;
        {   Décrémente le niveau du crayon (initialement 0). Le
    c   r   a   y   o   n                                   n   e
            trace que si son niveau est >= 0 }
    procedure ShowPen;
        {   Incrémente le niveau du crayon }
    procedure GetPen( var pt : Point );
        {   Retourne les coordonnées locales courantes du crayon }
    procedure GetPenState( var pnState : PenState );
        {   Retourne l'état du crayon dans pnState }
    procedure SetPenState( pnState : PenState );
        {   Restaure l'état du crayon à pnState }
    procedure PenSize( width, height : Integer );
        {   Définit la largeur et la hauteur courante du crayon }
    procedure PenMode( mode : Integer );
        {   Définit le mode de tracé du crayon }
    procedure PenPat( pat : Pattern );
        {   Définit le motif de tracé du crayon }
    procedure PenNormal;
        {   Restaure l'état normal du crayon (Taille 1x1, écrit en
    noir) }

{ Déplacements du crayon et tracés de segments }

    procedure MoveTo( h, v : Integer );
        {   Déplace le crayon au point (h,v) sans rien tracer }
```

```
      procedure Move( dh, dv : Integer );
           {    Déplace le crayon de dh horizontalement et de dv
      verticalement,
                sans rien tracer }
      procedure LineTo( h, v : Integer );
           {    Trace un segment de la position courante du crayon
      jusqu'en
                (h,v) et déplace le crayon en ce point }
      procedure Line( dh, dv : Integer );
           { Déplacement relatif du crayon de (dh,dv) avec tracé }

    { Texte  }

      procedure TextFont( font : Integer );
                      {    Choix de la police courante de caractères
(Police du système =
                           0; NewYork = 2; Geneva = 3; Monaco = 4;
Times = 20; Helvetica =
                           21; Courier = 22; Symbol = 23 }
      procedure TextFace( face : Style );
           {    Choix du style de la police courante de caractères }
      procedure TextMode( mode : Integer );
           {    Choix du mode d'écriture (srcOr, srcXor ou srcBic) }
      procedure TextSize( size : Integer );
           {    Taille de la police courante de caractères }
      procedure SpaceExtra( extra : LongInt );
           {    Nombres de pixels à ajouter aux blancs d'une ligne (à
      d   e   s                                          f   i   n   s
                de justification de texte }
      procedure DrawChar( ch : char );
           {    Trace le caractère ch et déplace le crayon }
      procedure DrawString( s : Str255 );
           {    Trace la chaîne de caractères s et déplace le crayon }
      function CharWidth( ch : char ) : Integer;
           {    Largeur de ch (en pixels) }
      function StringWidth( s : Str255 ) : Integer;
           {    Largeur de la chaîne de caractères s (en pixels) }
      procedure GetFontInfo( var info : FontInfo );
           {    Retourne les informations sur la police de caractères du
                grafPort courant }

    { Calcul sur les points }

      procedure AddPt( src : Point; var dst : Point );
           {    dst := src + dst }
      procedure SubPt( src : Point; var dst : Point );
           {    dst := src - dst }
      procedure SetPt( var pt : Point; h, v : Integer );
           {    pt := (h, v) }
      function EqualPt( pt1, pt2 : Point ) : boolean;
           {    pt1=pt2 ? }
      procedure ScalePt( var pt : Point; fromRect, toRect : Rect );
           {    pt := (pt.h*largeur(toRect)/largeur(fromRect),
                   pt.v*hauteur(toRect)/hauteur(fromRect)) }
      procedure MapPt( var pt : Point; fromRect, toRect : Rect );
           {    Transforme pt selon une similitude définie par fromRect
      et
                toRect }
      procedure LocalToGlobal( var pt : Point );
```

```
        {     Conversion des coordonnées locales (de la fenêtre) aux
              coordonnées globales (de l'écran) }
    procedure GlobalToLocal( var pt : Point );
        {     Conversion des coordonnées globales (de l'écran) aux
              coordonnées  locales (de la fenêtre) }

{ Calculs sur les rectangles  }

    procedure SetRect( var r:Rect; left,top,right,bottom:Integer );
        {     r:=((left,top),(right,bottom)) }
    function EqualRect( rect1, rect2 : Rect ) : boolean;
        {     rect1=rect2 ? }
    function EmptyRect( r : Rect ) : boolean;
        {     r est vide ? }
    procedure OffsetRect( var r : Rect; dh, dv : Integer );
        {     ajoute dh (dv) aux coordonnées horizontales (verticales)
    de r }
    procedure MapRect( var r : Rect; fromRect, toRect : Rect );
        {     Transforme r selon une similitude définie par fromRect
    et
              toRect }
    procedure InsetRect( var r : Rect; dh, dv : Integer );
        {     Rétrécir r de 2*dh horizontalement et 2*dv verticalement
    }
    procedure UnionRect( src1, src2 : Rect; var dstRect : Rect );
        {     dstRect := src1 ∪ src2 }
    function SectRect( src1,src2:Rect; var dstRect:Rect ) : boolean;
        {     dstRect := src1 ∩ src2 }
    function PtInRect( pt : Point; r : Rect ) : boolean;
        {     pt ∈ R ?}
    procedure Pt2Rect( pt1, pt2 : Point; var dstRect : Rect );
        {     dstRect est le plus petit rectangle contenant pt1 et pt2
    }

{ Dessin de rectangles }

    procedure FrameRect( r : Rect );
        {     Tracer un cadre intérieur à r avec le crayon }
    procedure PaintRect( r : Rect );
        {     Peindre l'intérieur de r avec le crayon }
    procedure EraseRect( r : Rect );
        {     Peindre l'intérieur de r avec le motif d'arrière-plan
    (voir
              BackPat) }
    procedure InvertRect( r : Rect );
        {     Passe l'intérieur de r en vidéo-inverse }
    procedure FillRect( r : Rect; pat : Pattern );
        { Remplir r avec le motif pat (en mode patCopy) }

{ Dessin de rectangles à coins arrondis (ovWd:largeur, ovHt:hauteur
    du
    quart d'ellipse constituant les coins) }

    procedure FrameRoundRect( r : Rect; ovWd, ovHt : Integer );
    procedure PaintRoundRect( r : Rect; ovWd, ovHt : Integer );
    procedure InvertRoundRect( r : Rect; ovWd, ovHt : Integer );
    procedure EraseRoundRect( r : Rect; ovWd, ovHt : Integer );
```

```
  procedure FillRoundRect( r:Rect; ovWd,ovHt:Integer; pat:Pattern
  );

{ Dessin d'ellipses  (spécifiées par le rectangle circonscrit) }

  procedure FrameOval( r : Rect );
  procedure PaintOval( r : Rect );
  procedure EraseOval( r : Rect );
  procedure InvertOval( r : Rect );
  procedure FillOval( r : Rect;pat : Pattern );

{ Dessin d'arcs spécifiés comme suit :
```




```
}

  procedure FrameArc( r : Rect; startAngle, arcAngle : Integer );
  procedure PaintArc( r : Rect; startAngle, arcAngle : Integer );
  procedure EraseArc( r : Rect; startAngle, arcAngle : Integer );
  procedure InvertArc( r : Rect; startAngle, arcAngle : Integer );
  procedure FillArc( r : Rect; startAngle, arcAngle : Integer;
                                                  pat : Pattern);
  procedure PtToAngle( r : Rect; pt : Point; var angle : Integer
  );
       {     Retourne l'angle (en °) entre la verticale passant par
  le
             centre de r et une ligne joingnant pt au centre de r }

{ Divers }

  procedure GetPixel( h, v : Integer ) : boolean;
       {     Vrai ssi le pixel (v,h) est noir }
  procedure Random : Integer;
       {     Retourne un nombre pseudo-aléatoire compris entre
  -  M  a  x  i  n  t  -  1                          e  t
             Maxint }


{ Routines non conformes au Macintosh Pascal Technical Appendix [2]
  }

  procedure ResetDrawing;
       { Réinitialisation de la fenêtre Drawing }
  procedure DrawByte( c : Byte );
       {     Trace le caractère de code c et déplace le crayon }
  function ByteWidth( c : Byte ) : Integer;
       {     Largeur du caractère de code c (en pixels) }
  procedure DrawText( var textBuf : char; firstByte : Integer;
                                                 byteCount:Byte );
```

```
    {    Trace la chaîne de caractères codée dans textBuf et
d   é   p   l   a   c   e                                   l   e
         crayon }
function TextWidth( var textBuf : char; firstByte : Integer;
                                        byteCount : Byte ) : Integer;
    {    Longueur de la chaîne de caractères codée dans textBuf
(comptée
         en pixels) }
procedure SetRandSeed( theSeed : LongInt );
    { Réinitialisation de la génération de nombres pseudo-
aléatoires }
```

5.6.4.4 Limites à la compatibilité des programmes Pascal sur le Macintosh et sous Unix (***)

5.6.4.4.1 Identificateurs (***)

Contrairement à Macintosh Pascal, le compilateur `pc` distingue les majuscules des minuscules. Sous Macintosh Pascal, il convient donc d'être méthodique et vigilant sous peine d'un travail de conversion fastidieux lors du transfert sous Unix.

5.6.4.4.2 Nombres (***)

Une autre source d'incompatibilité entre le Macintosh Pascal et le `pc` de Unix tient à la représentation des nombres. En effet :

- Sous Unix, le type `Integer`, associé à la constante `Maxint`, correspond au type `integer`, associé à `maxint` (resp. identiques à `Integer` et `Maxint`) du Macintosh c'est-à-dire des entiers codés sur 16 bits. Cependant les entiers de type `Integer` en Macintosh Pascal ont des valeurs comprises entre `-Maxint` et `Maxint` tandis que les entiers de type `Integer` sous Unix ont des valeurs comprises entre `-Maxint-1` et `Maxint`.
- Sous Unix, le type `LongInt`, associé à la constante `Maxlongint`, est un alias du type `integer` (entiers codés sur 32 bits) compatible avec le type `LongInt` du Macintosh.

Le type `real` est le même sur les deux machines (mais les représentations binaires de ces nombres différent). Les types `double`, `extended` et `computational` du Macintosh n'ont pas d'équivalent sur Unix. Les expressions à valeur réelle sont évaluées dans le type `extended` sur Macintosh, et dans le type `real` sous Unix, d'où une perte de précision!

Les procédures de gestion du curseur ne prennent effet que lorsqu'il se trouve dans la fenêtre `Drawing`.

5.6.4.4.3 Caractères et chaînes de caractères (***)

Les caractères sous Unix ont un code ASCII inférieur ou égal à 127 et les caractères Macintosh de code compris entre 128 et 255 ne sont donc pas représentables. C'est pourquoi, les sous-programmes `DrawChar` et `CharWidth` sont complétés par `DrawByte` et `ByteWidth`. Par exemple, pour afficher un `'é'`, il convient de remplacer l'instruction `DrawChar(chr(142))` de Macintosh Pascal par `DrawByte(142)`.

La représentation des chaînes de caractères diffère d'un Pascal à l'autre. Il n'est redéfini que le seul type `string` (et un alias, `Str255`) qui est presque équivalent au type `string` du Macintosh Pascal. Sous Unix, hormis le problème de codage des caractères, les blancs situés en fin de chaîne sont ignorés et une constante de type chaîne doit contenir au

moins deux caractères. Les constantes de zéro ou un caractère sont considérées de type `char` qui est incompatible avec le type `string`.

La procédure `WriteDraw` ne pouvant être définie sous Unix (sans intervenir dans le compilateur), il convient de remplacer les instructions `WriteDraw( ... );` par `writeln( DRAW, ... )`. Plus précisément, suite à `write( DRAW )`, tous les caractères reçus par le Macintosh jusqu'à atteindre un maximum de 255 ou jusqu'à réception d'un retour-chariot (`chr(13)`) sont traités de manière équivalente à un `DrawString`.

Les procédures `StringOf` et `ReadString` du Macintosh Pascal n'ont pas d'équivalent.

D'autre part, les procédures `DrawText` et `TextWidth` ont une interface modifiée. Le passage par valeur d'un pointeur sur le tampon de texte est remplacé par le passage en variable du premier caractère du tampon.

Soit, par exemple :

```
Buffer : array[ 1..1000 ] of char;
```

les appels

```
DrawText( @Buffer, 100, 10 );
```

et

```
DrawText( @Buffer[1], 100, 10 );
```

qui sont licites et équivalents sous Macintosh Pascal, doivent, sous Unix, être transformés en :

```
DrawText( Buffer[1], 100, 10 );
```

5.6.4.4.4 Divers (***)

Sous Unix, la fonction `Random` (attention il existe aussi une fonction `random`) utilise le générateur aléatoire d'Unix . La variable `randSeed` peut être lue, mais les affectations du type :

```
randSeed := exp;
```

doivent être remplacées par des appels :

```
SetRandSeed(exp).
```

Les valeurs retournées par `TickCount` sont des multiples de 60 et retourne un temps absolu avec une précision de l'ordre de la seconde. L'unité reste le 60^{ième} de seconde.

Sous MacIntosh Pascal, les modalités de "clipping" et les effets de la procédure `ClipRect` ne sont pas conformes aux spécifications et des problèmes surviennent notamment après un `SetOrigin`. Ces problèmes sont corrigés par TGiX.

Enfin, pour obtenir l'équivalent des initialisations implicites de Macintosh Pascal, il est nécessaire de placer l'instruction `ResetDrawing` au début du programme principal.

5.6.5 Versions successives de TGiX (***)

Les versions successives de `TGiX` sont en libre service sur le VAX 8600 et vous pouvez en charger une sur disquette par la commande :

`loadTGiX` ***i***

dont il suffit de suivre les instructions. La commande :

TGiXman *i*

permet de lister les extensions supportées par cette version. Les commandes `loadTGiX` et `TGiXman`, utilisées sans argument, listent les versions disponibles et vous proposent de choisir parmi elles.

La rubrique `x.graphic` des `news` permet d'échanger remarques et idées à propos de ces outils. Pour y accéder, vous disposez des commandes **`readnews x.graphic`** et **`postnews`**.Vous serez ainsi avertis de la mise à disposition des nouvelles versions. En retour, signalez les anomalies que vous pourriez constater et faites part de vos desirata pour les versions à venir.

6. REPERTOIRES SOUS UNIX

6.1 CREATION D'UN REPERTOIRE

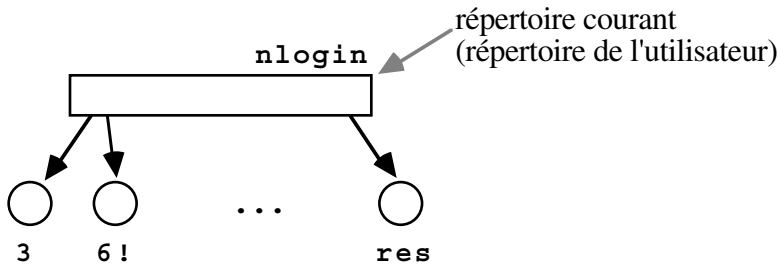
Quand le nombre de fichiers d'un utilisateur est grand, il devient absolument nécessaire de structurer cet ensemble de fichiers.

La structure des fichiers Unix est une structure arborescente dont les noeuds internes sont des *répertoires* (en anglais *directories*) et les feuilles sont des fichiers ordinaires (texte ou exécutable (c'est-à-dire contenant un programme machine)).

Pour l'instant, la structure des fichiers dont nous disposons est très simple:

```
%ls -F
3               a.out*          fact.p          fact1.p
6!              boucler.p       fact.x*         res
```

Elle se représente par le schéma :



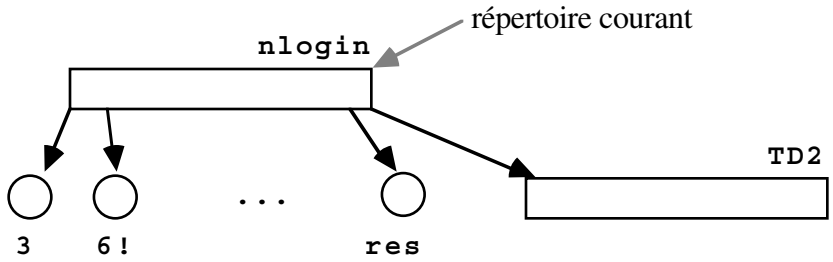
Il est possible de créer un nouveau répertoire de nom `TD2` (qui sera le fils du répertoire `nlogin`) par la commande (**m**ake **d**irectory):

```
%mkdir TD2
```

Nous disposons maintenant de la structure suivante:

```
%ls -F
3               TD2/            boucler.p       fact.x*
6!              a.out*          fact.p          fact1.p
res
```

(On reconnaît les répertoires par le fait que leur nom est suivi de "/". Le répertoire `TD2` nouvellement créé est vide).



6.2 REPERTOIRE COURANT

Toutes les commandes UNIX sont relatives au *répertoire courant* qui au début de la session est le répertoire de l'utilisateur portant son nom de bord `nlogin`**.**

On peut connaître le répertoire courant au moyen de la commande **`pwd`** (**p**rint **w**ork **d**irectory):

```
%pwd
/usr/users/cie/nlogin
```

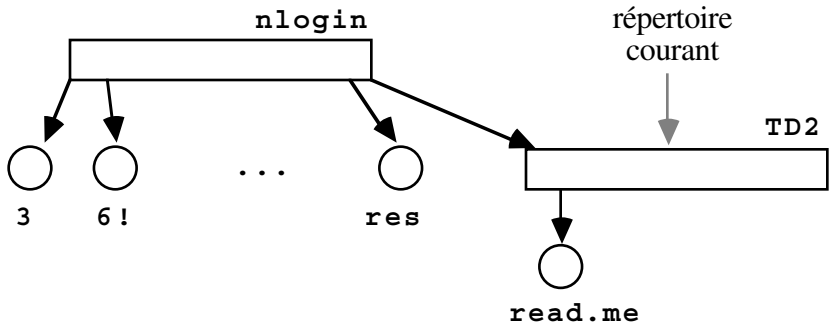
La commande **`cd`** (**c**hange **d**irectory) permet de changer le répertoire courant:

```
%cd TD2
%pwd
/usr/users/cie/nlogin/TD2
%ls
%
```

Il est utile d'associer à chaque répertoire un fichier `read.me` contenant un descriptif des fichiers du répertoire. On peut créer un tel fichier avec l'éditeur de texte `vi`, ce qui donne :

```
%ls -F
read.me
%pwd
/usr/users/cie/nlogin/TD2
%more read.me
TD2 : exercices du T.D. 2 d'Informatique
```

La structure des fichiers est alors la suivante :



La commande **`cd ..`** permet de remplacer le répertoire courant par son père:

```
%cd ..
%pwd
/usr/users/cie/nlogin
%ls -F
3               TD2/            boucler.p       fact.x*
6!              a.out*          fact.p          fact1.p
res
```

6.3 DESIGNATION D'UN FICHIER

Pour désigner un fichier, on utilise simplement son nom s'il appartient au répertoire courant. Sinon, il doit appartenir à un descendant du répertoire courant et l'on utilise un *chemin d'accès relatif* de la forme:

```
répertoire_1/.../répertoire_n/nfichier
```

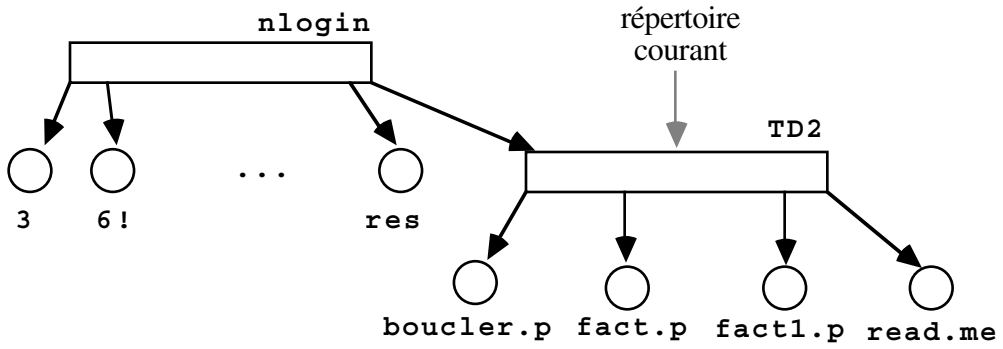
où `répertoire_1` appartient au répertoire courant, `répertoire_i+1` est un fils du `répertoire_i`, i=1, ... , n-1 et `nfichier` appartient à `répertoire_n`.

Par exemple, sachant que le répertoire courant est `nlogin`, les commandes de recopie de fichiers suivantes:

```
%pwd
/usr/users/cie/nlogin
%ls -F
3               TD2/            boucler.p       fact.x*
6!              a.out*          fact.p          fact1.p
res
%cp fact.p TD2/fact.p
%cp fact1.p TD2/fact1.p
%cp boucler.p TD2/boucler.p
```

changent la structure des fichiers comme suit:

```
%cd TD2
%pwd
/usr/users/cie/nlogin/TD2
%ls -F
boucler.p       fact.p          fact1.p         read.me
```



On peut maintenant supprimer les fichiers inutiles du répertoire de l'utilisateur. La commande **cd** (**c**hange **d**irectory) permet de ramener le répertoire courant sur le répertoire de l'utilisateur:

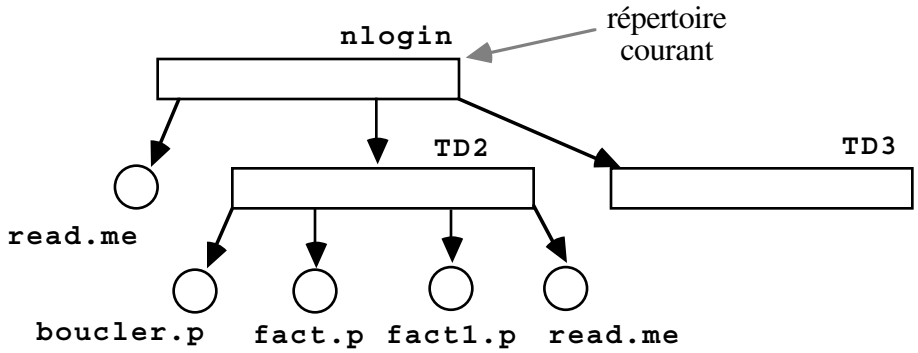
```
%cd
%pwd
/usr/users/cie/nlogin
%ls -F
3               TD2/            boucler.p       fact.x*
6!              a.out*          fact.p          fact1.p
res
%rm *
```

```
rm: remove 3? y
rm: remove 6? y
rm: TD2 directory
rm: remove a.out? y
rm: remove boucler.p? y
rm: remove fact.p? y
rm: remove fact.x? y
rm: remove fact1.p? y
rm: remove res? y
%ls -F
TD2/
```

Pour terminer, nous ajoutons un fichier `read.me` dans `nlogin` contenant un descriptif de ce répertoire et créons un répertoire `TD3` qui contiendra les exercices du troisième T.D. d'Informatique:

```
%mkdir TD3
%ls -F
TD2/     TD3/     read.me
```

La structure des fichiers est alors:



6.4 SUPPRESSION D'UN REPERTOIRE

Signalons enfin qu'un répertoire `nrep` peut être supprimé par la commande (**r**emove **d**irectory) **`rmdir nrep`** (à condition que le répertoire `nrep` soit vide):

```
%ls -F
TD2/    TD3/    read.me
%rmdir TD3
%ls -F
TD2/    read.me
%rmdir TD2
rmdir: TD2: Directory not empty
```

(Des compléments d'information sur les répertoires sont donnés en annexe § 9.1.5).

7. COMMUNICATION ENTRE UTILISATEURS DE UNIX (*)

7.1 TELECONFERENCE (**)

Il est possible d'initialiser un dialogue avec un correspondant (connu du système par son nom de bord `sonlogin`) par la commande :

`talk sonlogin`

Si le correspondant `sonlogin` n'est pas connecté, on obtient le message:

```
Your party is not logged in
```

Si le correspondant `sonlogin` est connecté (ce que l'on peut vérifier par les commandes **`users`**, **`who`**, **`fingers`**,...), il reçoit le message:

```
Message from Talk_Daemon@poly hh:mn ...
talk:  connection requested by votrelogin@poly
talk:  respond with:  talk votrelogin@poly
```

Votre correspondant doit répondre par:

`talk votrelogin`

Le dialogue peut alors s'établir simultanément, les textes qui sont tapés étant présentés séparement dans deux parties distinctes de l'écran, la partie supérieure étant réservée à l'appelant.

Pour mettre fin à la conversation taper CTRL `c`.

Sous `vi`, CTRL `l` permet de réimprimer proprement un écran qui serait perturbé par une demande de dialogue.

Pour éviter d'être dérangé, vous pouvez interdire la réception de messages par la commande :

`mesg n`

Dans ce cas, un interlocuteur essayant d'engager le dialogue avec vous recevra le message :

```
Permission denied
```

Vous pouvez ré-autoriser la réception de messages (option par défaut) par :

`mesg y`

Pour savoir si votre interlocuteur vous autorise à lui écrire, faites :

`finger sonlogin`

Le message `messages off` apparaît après son nom de bord `sonlogin` s'il a interdit les communications.

7.2 COURRIER ELECTRONIQUE LOCAL (*)

7.2.1 ENVOI DE COURRIER (*)

Vous pouvez envoyer un courrier électronique à un (ou des) correspondant(s) de nom de bord `nlogin...` par la commande :

`mail nlogin...`

Il apparaît alors sur l'écran :

```
Subject :
```

et vous devez donner le sujet du courrier puis le texte en autant de lignes que nécessaire. Si vous tapez deux CTRL `c`, la commande est interrompue, votre courrier ne sera pas expédié et le texte du courrier que vous étiez en train de taper se retrouve dans un fichier `dead.letter`. Sinon, il faut terminer le texte du courrier par CTRL `d` (ou par un point) au début d'une ligne.

Le texte :

```
EOT
%
```

apparaît sur l'écran, ce qui signifie que le message a été transmis à votre correspondant. Si le correspondant est inconnu votre courrier vous est retourné. Notez que votre correspondant peut être vous-même, le courrier électronique servant alors d'aide-mémoire.

7.2.2 AVIS DE RECEPTION (*)

Vous êtes averti que vous avez reçu du courrier électronique par le message:

```
You have mail.
```

au début d'une session. Si quelqu'un vous envoie du courrier pendant que vous êtes connecté, vous ne le saurez donc que lors de la prochaine session de travail, à moins que par une commande :

```
biff y
```

vous n'ayez demandé au système d'être prévenu lorsque du nouveau courrier vient de vous être envoyé (c'est l'option par défaut). La commande :

```
biff n
```

a l'effet contraire.

7.2.3 LECTURE DU COURRIER (*)

La commande **`mail`** permet de traiter le courrier reçu dans la boîte aux lettres. La réponse `No mail for nlogin` indique que la boîte aux lettres est vide. Sinon la réponse est une liste de messages avec, pour chacun d'eux, un numéro d'ordre, le nom de l'expéditeur, la date et l'heure d'expédition et le sujet. Ensuite le caractère d'incitation `&` apparaît en début de ligne, ce qui permet de traiter le courrier par les commandes suivantes :

`?`	pour obtenir un résumé des commandes,
`n`	pour afficher le message *`n`* (*`n`* est un numéro de message ou *`n1 n2`* … pour les messages *`n1 n2`* …, *`n1-n2`* pour les messages *`n1`* à *`n2`*, `*` pour tous les messages et `$` pour le dernier),
RETURN	pour afficher le message suivant,
`-`	pour afficher le message précédent,
`h`	pour obtenir la liste des messages,
`d` ***`n`***	pour effacer le message *`n`*,
`u` ***`n`***	pour annuler l'effet de la commande précédente,
`pre` ***`n`***	pour conserver le message ***`n`*** dans la boîte aux lettres,
`R` ***`n`***	pour répondre à l'expéditeur du message de numéro *`n`* (sans oublier de terminer le texte par CTRL `d` ou par un point en début de ligne),
`r` ***`n`***	pour répondre à l'expéditeur et à tous les récipiendaires du message ***`n`***,
`co [`*`n`***`] nfichier`**	pour recopier le message courant (ou de numéro *`n`*) à la fin de `nfichier` (qui est créé s'il n'existe pas),

x quitte le traitement en cours du courrier sans rien changer au contenu de la boîte aux lettres,

q quitte le traitement du courrier, les messages non examinés ainsi que les messages examinés mais préservés (par la commande `pre`) sont conservés dans la boîte aux lettres, les messages effacés (par la commande `d`) sont détruits, les messages examinés mais ni détruits ni préservés sont recopiés dans la corbeille à courrier (appelée `mbox`).

Les lettres contenues dans la corbeille à courrier `mbox` peuvent êtres traitées de la même façon par la commande :

`mail -f`

7.2.4 TRANSFERT DE FICHIERS PAR COURRIER ELECTRONIQUE (*)

Il est possible d'expédier un fichier `nfichier` à un correspondant `nlogin` par :

`mail nlogin < nfichier`

Le correspondant peut alors ranger les informations contenues dans `nfichier` dans le fichier de son choix au moyen de la commande **`co`** de **`mail`**.

7.3 ENVOI DE COURRIER ELECTRONIQUE SUR DES MACHINES DISTANTES (***)

Le VAX 8600 est relié à d'autres ordinateurs internes à l'Ecole Polytechnique par le réseau Ethernet (comme par exemple les SPS 7/300). Il peut également communiquer avec d'autres réseaux extérieurs à l'Ecole Polytechnique via une liason UUCP avec L'INRIA:

- Exemple d'envoi de courrier sur le SPS 7-3 à travers Ethernet à l'utilisateur dupont (sachant que tous les élèves ont un numéro de compte identique à celui du VAX8600 sur les SPS7/300):

```
%mail sps7-3\!dupont
```

- Exemple d'envoi de courrier à l'utilisateur dupont résidant à l'université de Columbia:

```
%mail inria\!cernvax\!CUCCFA.BITNET\!dupont
```

Contacter l'équipe système pour obtenir plus de précisions concernant les différentes adresses électronique en France ou à l'étranger.

7.4 NOUVELLES (***)

Le VAX 8600 reçoit régulièrement des nouvelles (`news`) du monde entier concernant des sujets très divers classés par thèmes (`newsgroups`). Utiliser la commande `newsgroups` pour connaitre la liste des groupes, la commande `readnews` pour lire les nouvelles et la commande `postnews` pour envoyer des nouvelles. Ne poster des nouvelles que pour les groupes commençant par x (x.general, x.graphic, x.mac, x.sps7,…) pour éviter que la nouvelle ne soit diffusée à l'extérieur (tout abus entrainerait l'exclusion de l'Ecole de ce réseau d'information).

8. TRAVAUX D'ARRIERE-PLAN ET DIFFERES (**)

8.1 TRAVAUX D'ARRIERE-PLAN (**)

Un utilisateur peut faire exécuter plusieurs programmes ou commandes en même temps, l'un s'exécutant de manière conversationnelle sur l'avant-scène, les autres en arrière-plan. On fait suivre pour celà la commande par **&**:

```
cmd &
```

ce qui exécute la commande `cmd` en arrière plan (et permet de continuer normalement sur l'avant-scène). Un message `[i] nnnn` donne le numéro *i* et l'identificateur *nnnn* du travail exécutant la commande.

Un programme s'exécutant en arrière-plan s'arrête s'il doit lire des données sur le clavier. On est prévenu à l'apparition du prochain caractère d'incitation `%` par un message de la forme:

```
[i]   + Stopped (tty input) cmd
```

Pour l'éviter, il est toujours possible de rediriger les entrées sur un fichier créé préalablement. On peut également reprendre le travail en conversationnel par:

```
%i
```

ce qui ramène le travail d'arrière plan numéroté *i* sur l'avant-scène.

Il est alors possible de taper les données sur le clavier, puis de ramener le travail en arrière-plan par:

CTRL **z**

qui arrête la commande en cours (en répondant `^Z Stopped`) puis par:.

```
bg
```

qui reprend son exécution en arrière-plan.

Quand un travail s'exécutant en arrière-plan doit écrire sur l'écran, il a le même comportement que pour les entrées, l'écran affichant un message:

```
[i] + Stopped (tty output) cmd
```

On peut connaître à tout moment l'état des travaux d'arrière-plan par la commande :

```
jobs   -l
```

qui donne le numéro *i*, l'identificateur *nnnn*, l'état et la commande exécutée par les travaux d'arrière plan de l'utilisateur.

Les travaux d'arrière-plan peuvent être arrêtés et repris à tout moment :

stop %*i*	arrête le travail d'arrière plan numéroté *i*,
%i &	reprend le travail numéroté *i* en arrière plan,
%i	reprend le travail numéroté *i* en avant-scène.

Quand un travail d'arrière-plan boucle ou produit des erreurs, on peut le supprimer par:

kill %*i*	supprime le travail d'arrière plan numéroté *i*,
kill *nnnn*	idem, en donnant l'identificateur du travail,
kill -9 %i	idem, si le travail ne veut pas mourrir.

S'il reste des travaux d'arrière-plan quand on fait `logout`, on obtient le message:

```
There are stopped jobs.
```

Il faut tuer ces travaux par:

```
kill %i
```
avant de terminer la session de travail.

8.2 TRAVAUX DIFFERES (***)

Un utilisateur peut lancer un travail `nfichier` à exécuter à une date `mmm jj` (où mmm est une abréviation du mois en Anglais) et une heure `hhmn` données (de préférence la nuit pour les travaux gourmants en temps de calcul) en utilisant la commande:

```
at hhmn mmm jj nfichier
```

Par exemple pour lancer `a.out` le 31 Dec à 22H 30, taper:

```
at 2230 Dec 31 a.out
```

Faire `man at` pour obtenir plus d'informations.

9. ANNEXES (*)

9.1 ABREGE DES COMMANDES UNIX (*)

9.1.1 CONVENTIONS ET COMMANDES UNIX D'INTERET GENERAL (*)

Conventions:

x	frappe de la touche *x*.
X	enfoncer SHIFT et taper *x*.
***x**	le caractère spécial ***x*** = BACKSPACE, @, \, <, >, *, ?, |, & est transmis tel quel.
CTRL ***x***	enfoncer CTRL et taper *x*.
BACKSPACE	efface le dernier caractère.
@	efface la ligne.
CTRL **s**	suspendre temporairement l'exécution de la commande qui produit la sortie sur l'écran.
CTRL **o**	supprimer la sortie sur l'écran (sans arrêter la commande qui la produit).
CTRL **q**	reprendre (l'exécution de la commande qui produit) la sortie sur l'écran.
CTRL **c**	arrêter définitivement l'exécution du programme ou de la commande en cours.
CTRL **d**	fin de fichier.

Nom de fichier (ou de répertoire):

nfichier	si le fichier *nfichier* est dans le répertoire courant.
répertoire_1/.../répertoire_n/nfichier	si *nfichier* est dans un répertoire descendant du répertoire courant.
/répertoire_1/.../répertoire_n/nfichier	désignation absolue à partir de la racine de l'arborescence des fichiers Unix.
.	désigne le répertoire courant.
..	désigne le père du répertoire courant.

Abbréviations (dans un nom de fichier, répertoire,…):

?	n'importe quel caractère.
*****	n'importe quelle chaîne de caractères, même vide.
[...]	n'importe quel caractère dans la liste ... (où *c1-c2* désigne l'ensemble des caractères entre *c1* et *c2* dans l'ordre des codes ASCII).

Commandes:

login *nlogin*	début de session sous le nom de bord *nlogin*.

date	imprime la date et l'heure.
cal [**m**] [**a**]	calendrier du mois m=1..12 de l'année a=1...9999.
clear	nettoie l'écran.
whoami	nom de bord.
tty	nom du terminal.
who am I	nom de bord, nom du terminal, date et heure.
groups	groupe (i.e. compagnie) du `nlogin`.
users	nom de bord des utilisateurs courants.
who	idem avec nom du terminal, date et heure.
w	idem avec description de l'activité courante des utilisateurs.
finger	comme `who` avec le nom en clair, téléphone,...
chfn ***nlogin***	pour changer les renseignements donnés par `finger` pour l'utilisateur portant le nom de bord *`nlogin`*.
whatis ***cmd***	description de la commande *`cmd`* en une ligne.
man ***cmd***	description complète de *`cmd`*.
prman [**-L**] ***cmd***	impression de la description complète de *`cmd`* sur l'imprimante rapide [l'imprimante laser] du Vax 8600.
man -k ***mot-clé***	description des commandes ayant rapport avec le *`mot-clé`*.
apropos ***mot-clé***	idem.
yes	boucle en imprimant `y`.
passwd	changement de mot de passe.
logout	fin de session.
cmd **<** ***nfichier***	prendre *`nfichier`* comme entrée standard de *`cmd`*.
cmd **>** ***nfichier***	prendre *`nfichier`* (qui n'existe pas) comme sortie standard de *`cmd`*.
cmd **>!** ***nfichier***	prendre *`nfichier`* (qui est détruit s'il existait) comme sortie standard de *`cmd`*.
cmd **>>** ***nfichier***	rediriger la sortie standard de *`cmd`* à la fin de *`nfichier`* (qui doit exister).
cmd **>>!** ***nfichier***	rediriger la sortie standard de *`cmd`* à la fin de *`nfichier`* (qui est créé s'il n'existait pas).
cmd1 **;** ***cmd2***	exécuter *`cmd1`* puis *`cmd2`*.
cmd1 **&&** ***cmd2***	exécuter *`cmd1`* puis *`cmd2`* s'il n'y a pas d'erreur quand *`cmd1`* s'exécute.
cmd1 **||** ***cmd2***	exécuter *`cmd1`* puis *`cmd2`* s'il y a une erreur quand *`cmd1`* s'exécute.
cmd1 **|** ***cmd2***	exécuter *`cmd1`* puis *`cmd2`* en prenant pour entrée standard de *`cmd2`* la sortie standard de *`cmd1`***.**
time ***cmd***	exécuter la commande *`cmd`* puis indiquer son temps d'exécution (en secondes).
script	garder dans le fichier `typescript` tout ce qui va être tapé sur le terminal. Prend fin par CTRL `d` (le fichier `typescript` peut ensuite être imprimé pour garder une trace écrite de la session de travail).
more /usr/lib/whatis	liste complète des commandes Unix avec une ligne d'explication pour chacune d'entre-elles.
man csh	description de l'interpréteur de commandes `csh`.

9.1.2 ABREGE DES COMMANDES DE L'EDITEUR `vi (**)`

- Invocation de l'éditeur vi pour le fichier de nom *`nfichier`*:

vi ***nfichier***

- Fin de l'édition:

ZZ	avec modification de *nfichier*.
:q!	sans modification de *nfichier*.

- Sauvegarde du texte (en cours d'édition):

:w	dans *nfichier*.
:w *nfichier'*	dans *nfichier'* sans modifier *nfichier*.

- Déplacement de la fenêtre:

CTRL **f**	en avant, d'une page.
CTRL **d**	en avant, d'une demi-page.
CTRL **b**	en arrière , d'une page.
CTRL **u**	en arrière, d'une demi-page.
z.	centrer la fenêtre à la ligne courante.
CTRL **l**	regénérer l'écran.
CTRL **y**	ajouter une ligne en haut de la fenêtre.
CTRL **e**	ajouter une ligne en bas de la fenêtre.
:1	placer la fenêtre au début du fichier.
:*n*	centrer la fenêtre sur $n^{ème}$ ligne.
:$	placer la fenêtre en fin de fichier.

- Déplacement du curseur:

M	au milieu de l'écran.

-- En arrière:

H	en haut, à gauche de l'écran.
{	au début du paragraphe courant (un paragraphe se termine par une ligne blanche).
(	au début de la phrase courante (une phrase se termine par une ligne blanche ou par **.**, **!** ou **?** suivis de deux blancs ou d'une ligne blanche).
k	sur la ligne précédente, en même colonne.
-	au début de la ligne précédente.

-- En avant:

L	en bas, à gauche, de l'écran.
}	à la fin du paragraphe courant.
)	au début de la phrase suivante.
j	sur la ligne suivante, en même colonne.
RETURN	au début de la ligne suivante.
:$	à la fin du fichier.

-- A droite:

SPACE ou **l**	à droite d'un caractère.
w	à droite au début du mot suivant.
e	à droite à la fin du mot courant.

$	à la fin de la ligne.

-- A gauche:

BACKSPACE ou **h**	à gauche d'un caractère.
b	à gauche au début du mot courant.
0	(zéro) au début de la ligne.

- Recherche (circulaire) d'un motif …

/…	en avant, après le curseur.
?…	en arrière, avant le curseur.
N	répéter la recherche en arrière.
n	répéter la recherche en avant.

(le motif … est une suite de caractères décrite comme suit (les conventions ne sont pas exactement les mêmes que sous l'interpréteur de commandes `csh`):

c	représente le caractère *c* (non ^ . $ [* \),
*****c*	représente le caractère *c* (qui est ^ . $ [* ou \),
^	représente le début de la ligne,
$	représente la fin de la ligne,
.	représente un caractère quelconque,
c*	représente une suite (la plus longue possible de 0 ou plusieurs caractères *c* (quelconques si c est .),
[...]	n'importe quel caractères dans la liste **...**,
[*c1*-*c2*]	n'importe quel caractère dont le code ASCII est compris entre ceux de *c1* et *c2*.

- Insertion de texte…:

i...ESC	à gauche du curseur.
a...ESC	à droite du curseur.
O...ESC	sur une nouvelle ligne au dessus du curseur.
o...ESC	sur une nouvelle ligne en dessous du curseur.
I...ESC	au début de la ligne.
A...ESC	à la fin de la ligne.
:r *nfichier'*	du fichier *nfichier'* à la fin du fichier *nfichier*.
:.r *nfichier'*	du fichier *nfichier'* après la ligne courante.

- Suppression de texte:

x	supprime le caractère courant.
dw	supprime le mot courant.
dd	supprime la ligne courante.
***n* dd**	supprime *n* lignes à partir de la ligne courante.
D	supprime le reste de la ligne à partir du caractère courant.

- Remplacement de texte:

r *c*	remplace le caractère courant par le caractère *c*.
R...ESC	remplace le caractère courant par le texte ...
cw...ESC	remplace le mot courant par le texte ...
C...ESC	remplace le reste de la ligne par le texte ...

`~`	change le caractère courant de minuscule en majuscule et vice-versa.
`:s /`*`…1…`***`/`***`…2…`***`/`**	remplace la première occurence du texte *…1…* dans la ligne courante par le texte *…2…* (`&` est une abbréviation de *…1…* dans *…2…*).
`:s /`*`…1…`***`/`***`…2…`***`/g`**	idem, pour toutes les occurrences de *…1…* dans la ligne courante.
`:`*`n1,n2`***` s /`***`…1…`***`/`***`…2…`***`/`**`[g]`	idem, pour les lignes *n1* à *n2* (*n1* et *n2* peuvent être un numéro de ligne, . (pour la ligne courante), `$` (pour la dernière ligne) ou combinés avec un déplacement de i lignes en avant (n+*i*) ou en arrière (n-*i*)).
`&`	répète la dernière commande **`s`**.

- Jonction de lignes:

`J`	joint la ligne suivante à la ligne courante.

- Coupure de lignes:

`i` RETURN ESC	coupe la ligne courante à gauche du curseur.
`a` RETURN ESC	coupe la ligne courante à droite du curseur.

- Duplication ou déplacement d'une partie de texte:

a)	**`dd`**	effacer la ligne courante.
	`n` **`dd`**	effacer *n* lignes à partir de la ligne courante.
	`Y`	marquer la ligne à dupliquer.
	`n` **`Y`**	marquer la première des *n* lignes à dupliquer.
b)	déplacer le curseur en nouvelle position.	
c)	**`P`**	insérer le texte effacé ou marqué au-dessus de la ligne du curseur.
	`p`	insertion en-dessous de la ligne du curseur.

- Duplication ou déplacement de plusieurs parties de texte:

a)	**`" `*`a`*` [`*`n`*`] Y`**	marquer la ligne de texte [le début des *n* lignes de texte] appelée(s) *a*, où *a* ∈ {a,...,z}.
	`" `*`a`*` [`*`n`*`] dd`	idem, en supprimant les lignes de texte marquées.
b)	déplacer le curseur en nouvelle position.	
c)	**`" `*`a`*` P`**	insérer le texte appelé *a* au-dessus du curseur.
	`" `*`a`*` p`	insertion de *a* en-dessous de la ligne du curseur.

- Répétition de commande:

`n cmd`	répéter la commande *cmd* *n* fois.

- Numérotation des lignes:

`:set nu`	numéroter les lignes.
`:set nonu`	enlever la numérotation.
`:.=`	numéro de la ligne courante.
`:=`	nombre de lignes dans le fichier.

- Annulation de l'effet d'une commande:

u	de la dernière commande.
U	restauration de la ligne courante avant les derniers changements.

- Exécution d'une commande UNIX:

:!*cmd*	exécute la commande UNIX *cmd* et reprend l'édition en cours.

9.1.3 MISE EN OEUVRE D'UN PROGRAMME PASCAL SOUS UNIX (*)

Le nom des fichiers contenant un programme PASCAL doit se terminer par le suffixe **.p**

- Commandes de compilation:

pc -C *nfichier.p*	compiler le programme PASCAL *nfichier.p* avec tests à l'exécution.
pc *nfichier.p*	compiler le programme PASCAL *nfichier.p* sans tests à l'exécution.
a.out	exécuter le dernier programme compilé (arrêt forcé par CTRL c).

- Références croisées:

pxref *nfichier.p*	références croisées du programme *nfichier.p*.
pxref -n *nfichier.p*	références croisées du programme *nfichier.p*, sans liste du programme.

- Commandes d'interprétation et de mise au point:

pix *nfichier.p*	interprétation du programme PASCAL *nfichier.p* (avec aide à la mise au point en cas d'erreur).
pix -p *nfichier.p*	idem, sans limite sur le nombre d'instructions exécutées.
pix -z *nfichier.p*	idem, avec profil d'exécution dans pmon.out.
pxp -z *nfichier.p*	profil de l'exécution précédente.
pi *nfichier.p*	préparer l'interprétation du programme *nfichier.p* dans le fichier obj.
px	interpréter le programme précédent.
pdx	mise au point du programme précédent à l'aide des commandes suivantes:
run	commencer l'exécution du programme,
run < *nf1* > *nf2*	idem, avec redirection des entrées-sorties,
cont	continuer l'exécution,
step	exécuter un pas du programme,
next	exécuter un pas du programme en sautant les appels de procédures et fonctions,
trace *l*	tracer l'exécution de la ligne *l*,
trace *p*	tracer les appels de la procédure *p*,
trace *v*	tracer les modifications de la variable *v*,
trace *e* at *l*	imprimer la valeur de l'expression *e* quand l'exécution passe à la ligne l,
trace	imprimer les lignes du programme qui sont exécutées,
stop at *l*	suspendre l'exécution à la ligne *l*,

stop in *p*	suspendre l'exécution quand la procédure *p* est appelée,
stop if *c*	suspendre l'exécution quand la condition *c* est vraie,
status	imprimer la liste des `trace` et `stop` en cours,
delete *n*	supprimer la `trace` ou le `stop` de numéro *n*,
assign *v e*	affecter la valeur de l'expression *e* à la variable *v*,
call *p*	appeler la procédure *p*,
where	écrire la liste des procédures en cours d'appel,
print *e*	écrire la valeur de l'expression *e*,
whatis *i*	écrire la déclaration de l'identificateur *i*,
which *i*	idem, y compris les déclarations englobantes,
dump [*nf*]	écrire la valeur de toutes les variables actives [dans le fichier *nf*],
list *l1, l2*	lister les lignes *l1* à *l2* du programme,
list *p*	lister la procédure ou fonction *p*,
edit *nfichier*	éditer le fichier *nfichier* (avec `vi`),
quit	quitter l'interpréteur PASCAL,
help	afficher un résumé des commandes ci-dessus à l'écran.

9.1.4 FICHIERS (*)

- Nom de fichier (ou de répertoire):

nfichier	relatif au répertoire courant.
répertoire_1/.../répertoire_n/nfichier	chemin d'accès relatif au répertoire cou
/répertoire_1/.../répertoire_n/nfichier	chemin d'accès absolu dans la structure de fichiers de Unix.
~	abbréviation du chemin d'accès absolu au répertoire *nlogin* de l'utilisateur.

- Commandes:

-- Liste des fichiers:

ls	liste des fichiers du répertoire courant.
ls -F	idem, * : exécutable, / : répertoire.
ls *nfichier*	indique si *nfichier* existe.
ll	donne le mode, ..., le nom de bord du propriétaire, le nombre de caractères, la date et l'heure de dernière modification et le nom des fichiers du répertoire courant.
du	nombre de kilo-octets occupés par le répertoire courant (noté **.**) et les répertoires qu'il contient.

-- Mode d'un fichier:

Le mode d'un fichier a la forme **drwxrwxrwx** où **d** indique que le fichier est un répertoire (sinon -). Les trois paquets **rwx** indiquent les droits respectifs de l'utilisateur, du groupe de l'utilisateur et des autres utilisateurs sur le fichier:

r	autorisation de lecture, sinon -,
w	autorisation de modification et suppression, sinon -,
x	autorisation d'exécution, sinon -.

-- Changement de mode d'un fichier:

chmod ***qui peut quoi nfichier...*** change les droits d'accès aux *nfichier...*,

qui	combinaison des lettres **u** (utilisateur) **g**(groupe) et **o** (autres),
peut	**+** pour donner et **-** pour retirer l'autorisation,
quoi	combinaison des lettres **r**, **w** et **x** (sans aucun blanc dans ***q****ui peut quoi*).

chmod go-rwx *	interdit l'utilisation des fichiers du répertoire courant par tout autre que le propriétaire et l'ingénieur système (option par défaut).

-- Listage de fichiers *nfichier...*:

more ***nfichier...***	listage par pages sur l'écran,
RETURN	une ligne de plus,
SPACE	une page de plus,
q	arrêt,
h	résumé des commandes.
cat > ***nfichier***	créer le fichier *nfichier* qui contiendra le texte tapé au clavier (terminer par CTRL d).
cat >> ***nfichier***	idem, en ajoutant le texte à la fin de nfichier.
cat ***nfichier...***	listage en continu des fichiers *nfichier...*.
cat -s ***nfichier...***	listage en continu sans les lignes blanches.
cat -n ***nfichier...***	listage en continu avec numérotation des lignes.
cat -n -b ***nfichier...***	listage en continu avec numérotation des lignes non blanches.
cat -v ***nfichier...***	listage en montrant les caractères non imprimables (comme le BACKSPACE,...).

-- Impression de fichiers:

--- sur l'imprimante rapide du Vax 8600:

lpr ***nfichier...***	impression en continu.
cat -n ***nfichier*** | **lpr**	idem, avec numérotation des lignes.
lpr -p ***nfichier***	impression par pages.

--- sur l'imprimante laser du Vax 8600:

elite ***nfichier...***	impression en caractères Elite.
courrier1 ***nfichier...***	impression en caractères Courrier1, format paysage.
courrier2 ***nfichier...***	impression en caractères Courrier2, format normal.

--- destruction d'un fichier dans le spoule (gestionnaire de file d'attente) de l'imprimante rapide ou l'imprimante laser du Vax 8600:

La commande **lpq** donne la liste des fichiers en attente d'impression sur l'imprimante rapide:

```
%lpq
lp is ready and printing
```

```
Rank   Owner      Jobs Files                                 Total Size
1st    dupont     707  Exemple.p                             5426 bytes
```

Utiliser de même **`lpq -Plaser`** pour connaître l'état des activités de l'imprimante laser (inactive dans l'exemple ci-dessous):

```
%lpq -Plaser
no entries
```

La commande **`lprm`** (ou**`lprm -Plaser`**) permet de retirer un fichier du spoule de l'imprimante:

```
%lprm 707
df707poly dequeued
```

seul le propriétaire du fichier ayant le droit d'exécuter cette opération.

-- Recopie et déplacement de fichiers:

`cp` ***`nf1 nf2`***	recopier le fichier *`nf1`* dans le fichier *`nf2`* (avec demande de confirmation si *`nf2`* existe).
`cp` ***`nf... nr`***	recopier les fichiers *`nf...`* dans le répertoire *`nr`* (avec demande de confirmation si *`nf`* existe).
`mv -i` ***`nf... nr`***	déplacer les fichiers *`nf...`* dans le répertoire *`nr`* (avec demande de confirmation si *`nf`* existe).

-- Changement du nom d'un fichier:

`mv -i` ***`nf1 nf2`***	change le nom du fichier *`nf1`* en *`nf2`* (avec demande de confirmation si *`nf2`* existe).

-- Suppression de fichiers:

`rm` ***`nfichier...`***	supprimer les *`nfichier...`* (demande de confirmation).

-- Transfert de fichiers entre le Vax 8600 et les disquettes du Macintosh:

`kermit -p -r`	recevoir un fichier du Macintosh (en conservant son nom).
`kermit -p -w -r`	recevoir un fichier du Macintosh (en conservant son nom *`nf`* ou sous le nom *`nf~1`*,... si ce fichier existe dans le répertoire courant).
`kermit -r -a` ***`nfichier`***	recevoir un fichier du Macintosh (sous le nom *`nfichier`*).
`kermit -s` ***`nfichier`***	envoyer *`nfichier`* au Macintosh.
`lecon` *`n`*	pour recevoir sur le Macintosh un fichier contenant un programme de la $n^{\text{ième}}$ leçon d'Informatique (la marche à suivre est indiquée).
`td` *`n`*	pour recevoir sur le Macintosh un fichier contenant un programme du corrigé des $n^{\text{ièmes}}$ TD d'Informatique (la marche à suivre est indiquée).

9.1.5 REPERTOIRES (*)

pwd	donne le chemin d'accès absolu au répertoire courant.
ls	liste des fichiers du répertoire courant.
ls -a	idem, y compris les fichiers système (. nom du répertoire courant, .. nom
ls -F	idem, (suivis de / pour les répertoires et de * pour les programmes mac
ls -l	idem, y compris les droits d'accès, le nom de bord du propriétaire, la taille en octets et la date de la dernière modification.
cd	le répertoire de l'utilisateur devient le répertoire courant.
cd ..	le père du répertoire courant devient le répertoire courant.
cd *nrep*	*nrep* devient le répertoire courant.
mkdir *nrep...*	créer les répertoires *nrep...* (relativement au répertoire courant).
rmdir *nrep...*	supprimer les répertoires *nrep...* (qui doivent être vides).
rm -r *nrep*	supprimer les fichiers et répertoires contenus dans *nrep* puis le répertoire *nrep* lui-même (avec demande de confirmation).
mv *nrep1 nrep2*	changer le nom du répertoire *nrep1* en *nrep2*.

9.1.6 COURRIER ELECTRONIQUE (**)

- Dépouillement du courrier:

mail	consultation de la liste des nouveaux messages en attente, (No mail si pas de courrier),
mail -f	consultation des messages de mbox,
? ou **help**	résumé des commandes,
h	pour obtenir la liste et le sujet des messages en attente (**+** (**-**) pour les 18 messages suivants (précédents)),
f [***n***]	sujet du message courant [de numéro *n*],
RETURN ou **+**	afficher le message suivant,
-	afficher le message précédent,
n	afficher le message de numéro *n*,
d [***n***]	marquer que le message courant [de numéro *n*] est à effacer en sortie,
u [***n***]	pour annuler l'effet de la commande précédente,
pre [***n***]	marquer que le message courant [de numéro *n*] est à préserver en sortie (au lieu de le recopier dans mbox). N'annule pas **d**,
r [***n***]	répondre à l'expéditeur et aux récipiendaires du message courant [de numéro *n*] en terminant par CTRL d ou un point en début de ligne,
R [***n***]	répondre à l'expéditeur du message courant [de numéro *n*],
s [***n***] ***nfichier***	sauvegarder le message courant [de numéro *n*] à la fin de *nfichier* (qui est crée dans le répertoire courant s'il n'existe pas),
co [***n***] ***nfichier***	idem sauf que le message n'est pas marqué 'à effacer en sortie',
chdir [***ndir***]	positionner le répertoire courant en *nlogin* [sinon en *ndir*],
m *nlogin...*	envoyer du courrier à *nlogin...*.

q	quitter `mail` en copiant les messages lus dans `mbox`, en laissant les messages non lus ou préservés en attente d'une prochaine lecture et en supprimant les messages marqués à effacer,
x	quitter sans rien changer.

- Envoi du courrier:

mail *nlogin* < *nfichier* envoyer *nfichier* à *nlogin* par courrier.

mail *nlogin*	envoyer un message à *nlogin*.
`Subject: ...`	entrer le sujet en une ligne,
...	taper le message,
.	terminer le texte du message par un point en première colonne,
`EOT`	

9.2 LE LOGICIEL `Kermit` D'EMULATION DE TERMINAL (***)

`Kermit` est un émulateur de terminal alpha-numérique VT 100 qui n'offre aucune possibilité graphique. Il permet de transférer des fichiers entre le Vax 8600 et les disquettes du Macintosh (y compris les applications comme `MacWrite`,…). Il peut être utilisé pour se connecter au Vax 8600 en même temps que l'on utilise l'interpréteur PASCAL du Macintosh.

Les menus de `Kermit` sont les suivants:

`Sortie`:
- `Sortie`: la seule option de ce menu permet de quitter `Kermit` tout en se déconnectant automatiquement du Vax 8600).

`Modes`:
- `Connection`: pour se connecter au Vax 8600 (qui répond `login`, `%`,…),
- `Emission`: pour envoyer un fichier au Vax 8600 (qui l'attend après une commande `kermit` (voir §3.4), la transmission est interrompue en cliquant sur la souris),
- `Reception`: pour recevoir un fichier du Vax 8600 (qui l'envoie après une commande `kermit` (voir § 3.7), la transmission est interrompue en cliquant sur la souris),
- `Paramètres`: pour régler les paramètres (pour notre installation: `Baud Rate` = 9600, `Bits Sent` = 8, `Stops Bits` = 1, `ON LINE`, `Parity` = `none`, `Xon/Xoff Flow` = `yes`, `tabs` en positions 1, 9, 17, 25,…).

`Options`:

`Image`,`MacWrite`, `Exécutable`, `Texte`: à positionner avant de transférer un fichier comme suit:

Vax 8600:	Options sélectionnées:	Modes:
fichier texte (programme PASCAL,…):		
`kermit -p n -r` Emission	`MacWrite` & `Text`	
`kermit -p n -s` *`nfichier`* Réception	`MacWrite` & `Text`	
application (MacWrite,…):		
`kermit -p n -i -r` Emission	`Image` & `Exécutable`	
`kermit -p n -i -s` *`nfichier`* Réception	`Image` & `Exécutable`	

9.3 CONSTITUTION D'UNE DISQUETTE D'EDITION DE TEXTES A PARTIR DE LA DISQUETTE `Terminal (***)`

Nous donnons ci-dessous la procédure à suivre pour créer une disquette contenant l'éditeur de textes `MacWrite` et permettant l'impression laser. `MacWrite` offre la possibilité de mettre le texte en pages. Ce texte peut également contenir des dessins (créés par

MacPaint ou MacDraw et insérés avec le presse-papiers ou ArtGrabber) ou des formules mathématiques (créées par MacEqn). C'est l'éditeur de texte que nous utilisons pour rédiger les polycopiés d'Informatique.

◊ La première étape consiste à **créer une disquette contenant le système d'exploitation** du Macintosh:

- insérer la disquette Terminal dans le lecteur interne,
- cliquer sur l'option Finder du MiniFinder,
- insérer une disquette vierge dans le lecteur externe, il apparaît une fenêtre de dialogue,
- cliquer sur Double-face pour initialiser la disquette en 800K,
- taper le nouveau nom WriteLaser de la disquette vierge,
- recopier les dossiers Dossier système et Impression (contenant les fichiers LaserWriter et LaserPrep) de Terminal sur WriteLaser,
- sélectionner l'option Redémarrer du menu Rangement.

◊ Il faut maintenant **recopier MacWrite sur WriteLaser**. Le plus simple est de procéder comme suit:

- insérer la disquette WriteLaser dans le lecteur interne puis sélectionner Finder,
- insérer une disquette contenant MacWrite (disquette disponible à la bibliothèque) dans le lecteur externe,
- recopier le fichier MacWrite en glissant son icône sur la disquette WriteLaser.

Sinon MacWrite (ainsi que d'autres applications du Macintosh) est disponible sur le Vax 8600 et peut être recopié avec TGiX (mais pas Versaterm qui n'installe pas correctement les applications):

- insérer la disquette Terminal dans le lecteur interne,
- se connecter au Vax 8600 à l'aide de TGiX,
- taper la commande:
  ```
  kermit -p n -i -s /usr/local/mac/macwrite
  ```
- sélectionner l'option Recevoir une archive du menu Transfert,
- une fenêtre apparaît, insérer la disquette WriteLaser dans le lecteur externe,
- donner le nom MacWrite au fichier à recevoir,
- vérifier que l'on a le nom de disquette WriteLaser (sinon sélectionner Lecteur) puis sélectionner Enregistrer,
- sélectionner l'option Quitter du menu Liaison,

◊ Il faut ensuite **installer le MiniFinder**:

- Redémarrer,
- insérer la disquette WriteLaser dans le lecteur interne,
- sélectionner l'option Finder du MiniFinder,
- sélectionner MacWrite (qui noircit), sélectionner l'option Utiliser le MiniFinder... du menu Rangement puis répondre Installer à la question de savoir s'il faut installer ou supprimer le MiniFinder,

◊ Enfin il faut **sélectionner l'imprimante LaserWriter** (ce qui ne peut se faire que sur un Macintosh connecté à cette imprimante):

- sélectionner Sélecteur du menu ,

- sélectionner `AppleTalk connecté`, l'icône de l'imprimante `LaserWriter`,
- terminer en fermant la fenêtre.

On procède de même pour créer des disquettes contenant d'autres logiciels comme `MacPaint` ou `MacDraw`. On peut également recopier le fichier `ImageWriter` sur la disquette pour l'impression avec les imprimantes à aiguilles .

10. REFERENCES ET BIBLIOGRAPHIE

[1] Hueras J., Macintosh Pascal Reference Manual, Apple Computer Inc., 1984.

[2] Hueras J., Macintosh Pascal Technical Appendix, Apple Computer Inc., 1984.

[3] Inside Macintosh, Vol. I,Addison-Wesley Pub. Co., 1985.

[4] Rifflet J. M., *La programmation sous Unix*, McGrawHill, 1986.

11. INDEX