Aide-mémoire commandes texte Unix

Les plus utiles, avec leurs options les plus courantes.

Filtres simples

`cat`		copie l'entrée sur la sortie, telle quelle.
	`-n`	numérote les lignes, à partir de 1
	`-b`	idem sans numéroter les lignes blanches
	`-s`	"squeeze" les lignes blanches consécutives
`head`		ne garde que le début (10 lignes par défaut)
	`-`*n*	garde *n* lignes
`tail`		ne garde que la fin (10 lignes par défaut)
	`-`*k* ou `-n` *k*	garde *k* lignes
	`+`*k* ou `-n +`*k*	garde à partir de la ligne *k*
`sort`		trie les lignes (alphabétiquement par défaut)
	`-r`	(*revert*) : ordre inverse
	`-n`	ordre numérique
	`-k` *n*	trie selon le *n*-ième champ de chaque ligne
	`-t '`*c*`'`	spécifie le séparateur de champ
	`-u`	(*unique*) : élimine les lignes consécutives identiques
	`-S` *size*	spécifie la taille de la mémoire vive utilisée (ex : 128M, 1G)
	`-T` *dir*	spécifie le répertoire temporaire utilisé (si `/tmp` est trop petit)
`tr`		"traductions" de caractères
	`'`*ch1*`' '`*ch2*`'`	remplace chaque caractère de *ch1* par le caractère correspondant dans *ch2*
	`-d '`*ch*`'`	(*delete*) : supprime tous les caractères de *ch*
	`-s '`*ch*`'`	"squeeze" tous les caractères consécutifs de *ch*
		caractères spéciaux :
		`\t` (tabulation), `\n` (*new line*), `\\` (*backslash*)
		chaînes spéciales : `a-z`, `A-Z`, `0-9` etc.
`cut`		découpe les lignes en "champs"
	`-f`*n*	conserve uniquement le *n*-ième champ
	`-f`*n*`,`*m*	idem champs *n* et *m*
	`-f`*n*`-`*m*	idem champs *n* à *m*
	`-d'`*c*`'`	délimiteur de champs, '`\t`' (TAB) par défaut
`uniq`		"squeeze" les lignes *consécutives* identiques
		(on applique souvent `sort` au préalable).
	`-c`	compte le nombre de lignes identiques

Recherche de motifs

Dans les commandes ci-dessous, l'*expression* recherchée peut être un simple mot ou une expression plus complexe (voir syntaxe dans la rubrique *regular expressions* de `man grep`).

Ces deux commandes fonctionnent en filtre mais `grep` peut aussi travailler avec des fichiers passés en paramètre.

`grep`		*`expression`* [*`fichier(s)`*]
		recherche les lignes contenant l'expression.
		Avec fichier(s) en paramètre,
		chaque ligne est précédée du nom du fichier
	`-c`	compte le nombre de lignes qui devraient être écrites
	`-v`	inverse le résultat (lignes qui ne contiennent pas l'expression)
	`-A` *`n`*	(*after*) : affiche les *n* lignes qui suivent chaque ligne sélectionnée
	`-B` *`n`*	(*before*) : idem lignes qui précèdent
	`-C` *`n`*	(*context*) : idem lignes autour
	`-f` *`file`*	lit les expressions (une par ligne) dans un fichier
	`-i`	(*ignore case*)
	`-n`	numérote les lignes
	`-w`	(*word*) : l'expression doit former un mot complet
		(un "mot" est constitué de lettres, chiffres ou *underscore*)
`sed`		`'s/`*`expression`*`/`*`remplacement`*`/g'`
		remplacement d'expressions.
		Sans le `/g` (*global*),
		seule la première occurrence de chaque ligne est remplacée.
		on peut aussi utiliser autre chose que le slash,
		pour plus de lisibilité, selon les expressions :
		`'s@`*`expression`*`@`*`remplacement`*`@g'`
		`'s#`*`expression`*`#`*`remplacement`*`#g'`
		`'s `*`expression remplacement`*` g'`
	`-n`	avec `/p` (éventuellement `/gp`),
		seules les lignes modifiées sont recopiées
	`-f` *`file`*	lit les commandes (une par ligne) dans un fichier

Autres

`wc`		(*word count*)
	`-c`	caractères
	`-w`	mots
	`-l`	lignes
`diff`		*f1 f2* lignes différentes entre deux fichiers. Si l'un des deux paramètres est `-`, c'est l'entrée standard
	`-b`	(*blank*) ignore les différences dans le nombre d'espaces
	`-B`	ignore les lignes blanches
	`-i`	(*ignore case*)
	`-q`	(*quiet*) indique seulement si les fichiers diffèrent
	`-y`	affichage côte-à-côte
`paste`		*fichiers* inverse de `cut` : la sortie contient une colonne pour chaque fichier. Si l'un des deux paramètres est `-`, c'est l'entrée standard
`join`		*f1 f2* jointure : fusionne les lignes de *f1* et *f2* selon une clef présente dans les deux. Les deux fichiers doivent être triés selon cette clef.
`zcat` *f*		décompresse un fichier `.gz` et l'écrit sur la sortie. Utile pour ne pas décompresser le fichier en entier sur le disque. Identique à `gzip -cd` et `gunzip -d`
`awk`		Lit un fichier ligne par ligne, découpe chaque ligne en champs et effectue une opération donnée en argument sur les champs de chaque ligne. Le premier champ est désigné par `$1`, le deuxième par `$2`, ... La ligne entière est désignée par `$0`. NR représente le numéro de la ligne courante. Les instructions sont passées en argument, entre apostrophes. Ex : awk '{if($1 > 2) print $0 ;}' Affiche toutes les lignes pour lesquelles le premier champ est supérieur à 2. Ex : awk '{sum+=$2 ; print sum/NR ;}' Affiche pour chaque ligne la somme de tous les deuxièmes champs des lignes précédentes, divisée par le nombre de lignes.