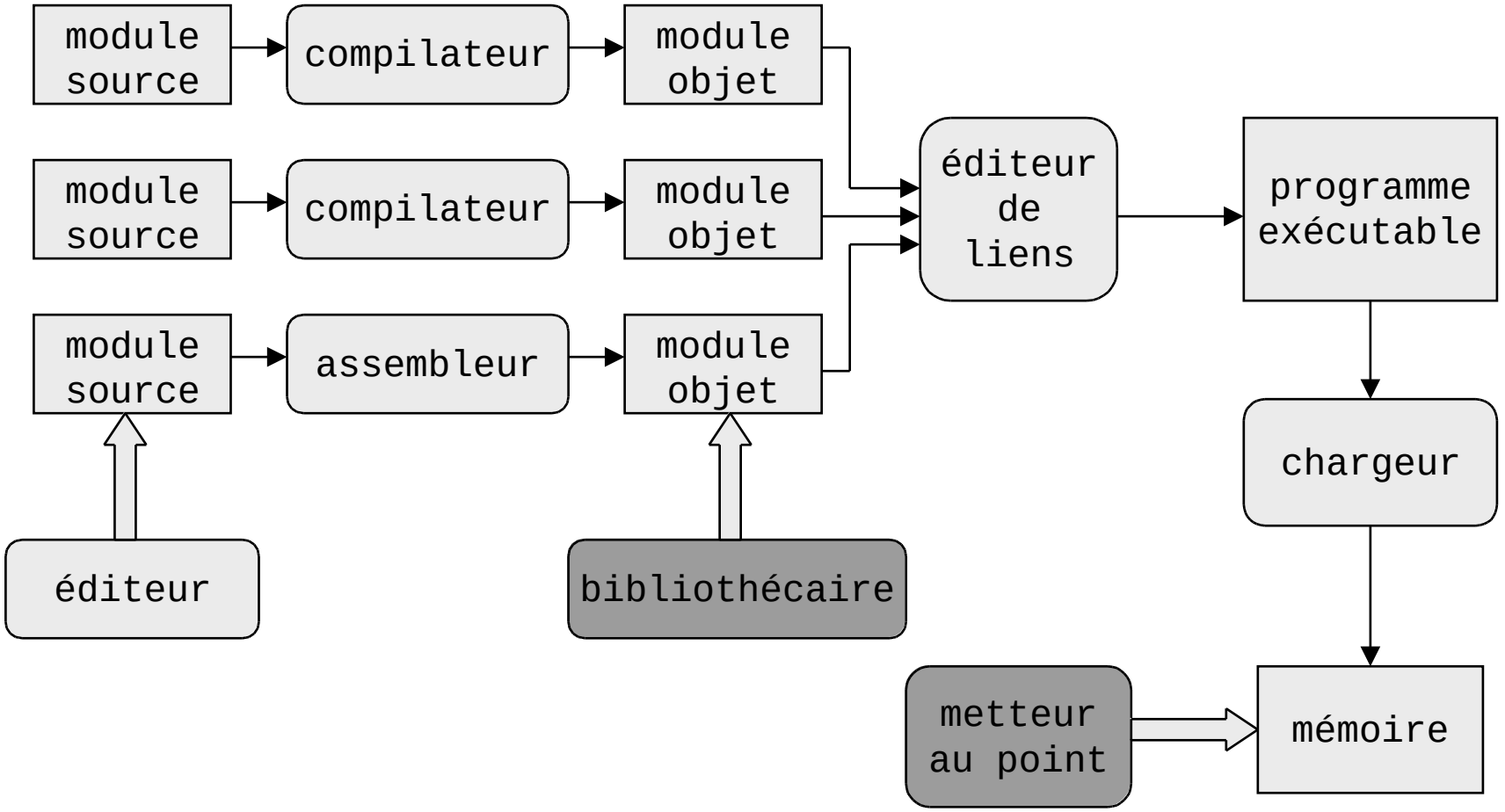
Autres outils de la chaîne de production de programme


module source
compilateur
module objet
module source
compilateur
module objet
module source
assembleur
module objet
éditeur de liens
programme exécutable
chargeur
éditeur
bibliothécaire
metteur au point
mémoire

Notion de trace

- Suivre l'évolution de l'état du programme
  - ♦ appels de sous programmes, paramètres, retour
  - ♦ déroulement des instructions, ruptures de séquences
  - ♦ valeurs successives des variables
  - ♦ état des données en certains points précis
- Compilation
  - ♦ instructions générées par le traducteur
  - ♦ difficulté de déterminer les bonnes options (ni trop, ni trop peu)
  - ♦ recompilation pour étendre ou supprimer la trace
- Édition de liens ou chargement => pas commode

  => *nécessité d'un outil interactif*

Point d'arrêt, reprise

- Notion de point d'arrêt
  - adresse d'emplacement contenant une instruction du programme où on désire que son exécution soit interrompue
  - but: permettre la consultation de l'état du programme à cet endroit
- Notion de reprise
  - permettre de poursuivre l'exécution d'un programme interrompu sur un point d'arrêt
- Notion de pas à pas
  - exécution du programme une instruction à la fois
    - niveau machine => instruction machine et non langage évolué
    - niveau langage évolué => où commence chaque instruction du langage?

Notion de metteur au point

- outil interactif qui permet:
  - mettre en route ou arrêter des options de trace
  - consulter et modifier des variables du programme
  - créer et supprimer des points d'arrêt
  - exécuter le programme en pas à pas
- metteur au point binaire
  - fournit ces fonctionnalités au niveau langage machine
    - variables => adresses binaires des emplacements
      format d'édition par décision de l'utilisateur
    - instructions => celles de la machine binaire
      avec éventuellement désassembleur (mnémonique)
  - nécessite une bonne connaissance de l'architecture de la machine

Metteur au point symbolique

- Fonctionnalités habituelles
  - mise en route et arrêt des traces, consultation et modification des variables création et suppression des points d'arrêts, exécution en pas à pas
- En plus:
  - consultation des appels de procédures en cours
  - consultation du fichier source
  - édition du fichier source (aide mémoire)
- Surtout: connaissance du langage source
  - identificateurs du texte source et adresses associées
  - déclarations et donc types des objets => format et notation pointée
  - correspondance instruction source -> emplacements mémoire
- Par le biais de tables
  traducteurs -> éditeur de liens -> metteur au point

Metteur au point multi-fenêtre

fenêtre de source:

point d'arrêt actuel

```
begin
    x := x + 2 * y;
  ➪ z := x * x + x /y;
    w := sqrt (z);
end;
```

fenêtre des variables:

```
x = 23.5
y = 1.23
x = 25.96
```

boutons de commande:

fenêtre de commandes:

```
print x
print y
print x
```

Mesures de comportement dynamique

- Rôle:
  - compter le nombre de fois où une suite d'instructions est exécutée
  - mesurer le temps passé à l'exécution dans une suite d'instructions
- Moyen:
  - option du traducteur (insertion d'instructions machines)
  - incrémentation d'un compteur avant l'exécution
  - appel au système avant et après l'exécution
- Résultats:
  - mémorisés dans un fichier et rattachés au source par un outil spécifique
- Intérêt:
  - optimiser l'exécution du programme là où on passe beaucoup de temps
- Exemples: en µs => instructions, en ms => sous-programmes

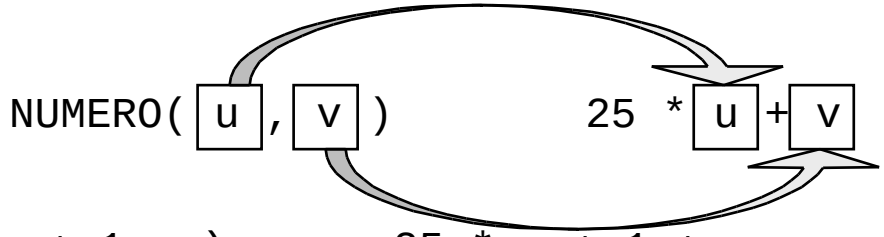
Préprocesseur et macrogénérateur(1)

- Paramétrage de textes sources:
  - texte unique => plusieurs versions de modules objets
- Exemple: préprocesseur C (et Pascal sur Unix, ...)

`#include "nom de fichier"`	inclusion du fichier à cet endroit
`#define TOTO 2456`	remplacement de la chaîne `TOTO` par la chaîne `2456`, partout
`#undef TOTO`	arrêt du remplacement de la chaîne `TOTO`
`#ifdef TOTO`	s'il y a un remplacement de `TOTO`, transmettre
`...`	les lignes qui suivent et supprimer celles
`#else`	après `#else`
`...`	sinon, supprimer celles qui suivent et transmettre
`#undef TOTO`	celles après `#else`

Préprocesseur et macrogénérateur(2)

- Exemple: `#define NUMERO(A,B) 25 * A + B`




```
NUMERO(x + 1, y)        25 * x + 1 + y
NUMERO((x + 1), y)      25 * (x + 1) + y
➪ #define NUMERO(A,B) 25 * (A) + (B)
```

- méta-langage: langage de commandes dans un texte source
- macrogénérateur: programme qui interprète ce métalangage pour générer un programme source
- Application:
  - requêtes de bases de données => appels sous programmes

Le Make (1)

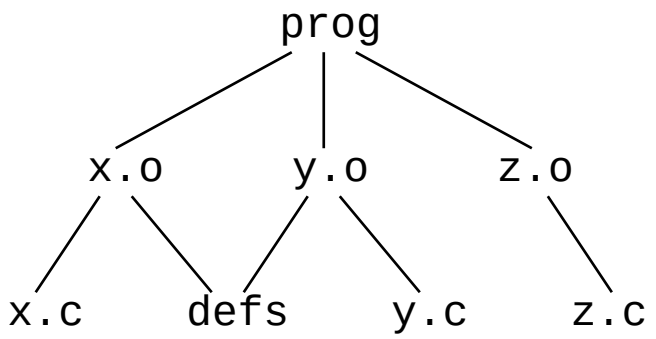
- Notion de graphe de dépendance

  On dit qu'un fichier A dépend d'un fichier B si B est utilisé pour construire A

- Si A dépend de B et date(A) < date(B) => A n'est pas à jour
- Exemple:
  - un module objet dépend du module source
  - un programme exécutable dépend des modules objets dont il est l'édition de liens

➪ Graphe de dépendance

`x.c` et `y.c` "incluent" `defs`



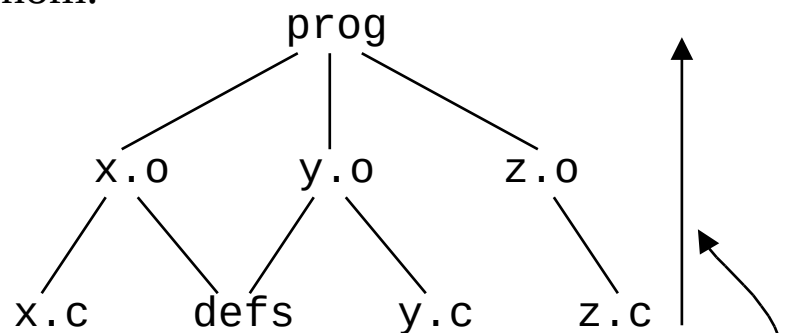
Le Make (2)

- Construction et exploitation du graphe de dépendance
- Définitions explicites dans un fichier paramètre (makefile)

```
prog: x.o y.o z.o
	ld x.o y.o z.o -ls -o prog
x.o y.o: defs
```

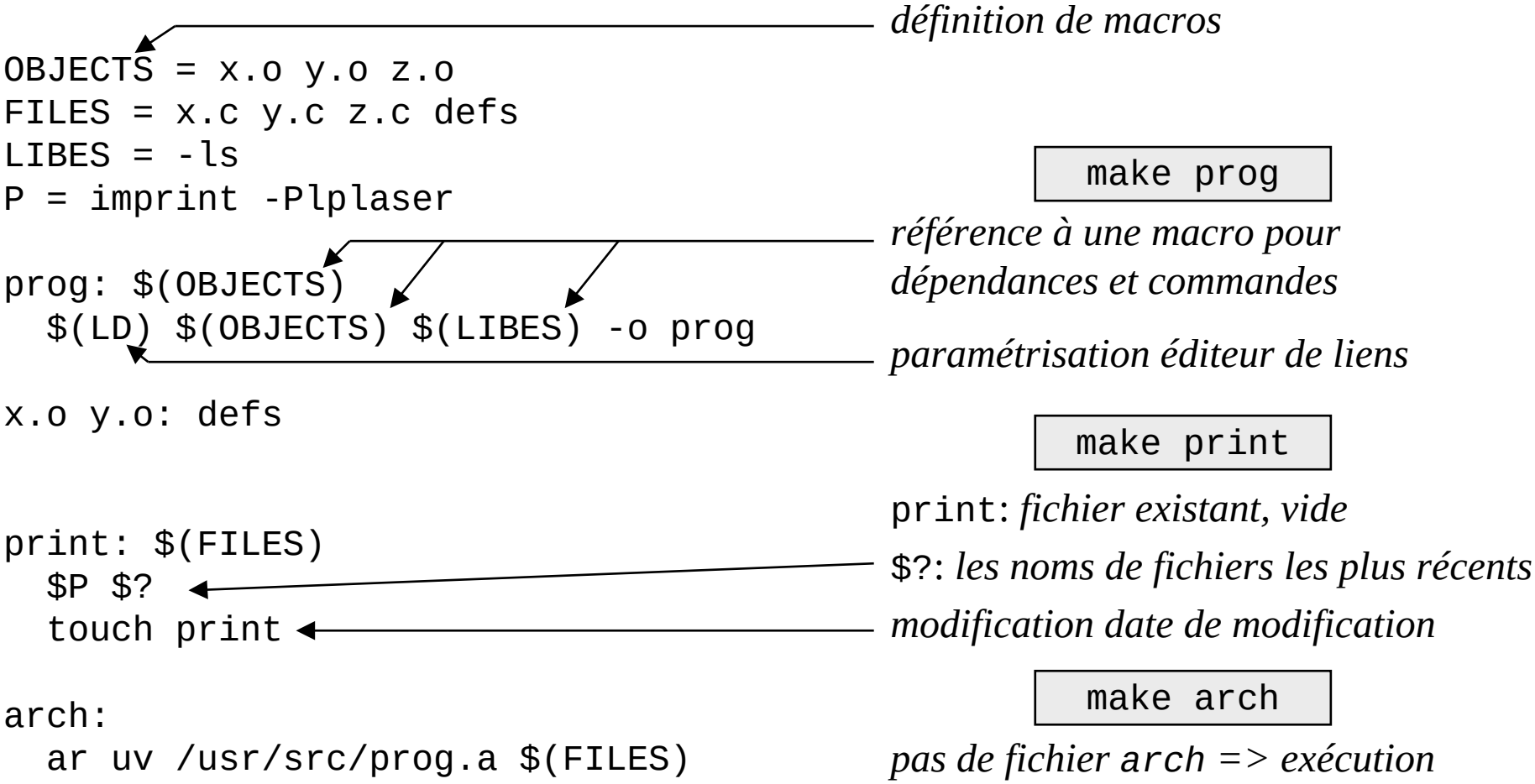
- Définitions implicites par suffixes
  - ♦ fichier "nom.o", recherche fichiers "nom.*"

```
x.o: x.c
	cc -c x.c
y.o: y.c
	cc -c y.c
z.o: z.c
	cc -c z.c
```



commandes de reconstruction

Le Make (3)


définition de macros
OBJECTS = x.o y.o z.o
FILES = x.c y.c z.c defs
LIBES = -ls
P = imprint -Plplaser
make prog
référence à une macro pour dépendances et commandes
prog: $(OBJECTS)
$(LD) $(OBJECTS) $(LIBES) -o prog
paramétrisation éditeur de liens
x.o y.o: defs
make print
print: fichier existant, vide
print: $(FILES)
$P $?
$?: les noms de fichiers les plus récents
touch print
modification date de modification
make arch
arch:
ar uv /usr/src/prog.a $(FILES)
pas de fichier arch => exécution

Outils complémentaires

- Archiveur
  - ♦ regrouper en un seul fichier plusieurs fichiers quelconques, en les compactant (gain 2 ou plus), avec conservation du découpage
- Différence
  - ♦ déterminer la suite de commande éditeur de texte qu'il faut appliquer à un fichier A pour obtenir le fichier B, à partir du contenu de ces fichiers
- Paragrapheur
  - ♦ mettre en forme standard un fichier source écrit dans un langage évolué
- Bibliothécaire
  - ♦ gérer les modules objets d'une bibliothèque, pour faciliter le travail de l'éditeur de liens

Il est préférable de faire un outil pour une activité automatisable plutôt que de réaliser l'activité à la main, car l'outil sera réutilisable.

Conclusions

- Metteur au point => outil de contrôle d'exécution d'un programme
  - trace: suivi de l'état
  - point d'arrêt: endroit où on force l'arrêt
  - reprise: poursuite d'exécution
  - pas à pas: exécution une instruction à la fois
- Mesures de comportement dynamique: amélioration temps exécution
- Préprocesseur, macrogénérateur : traitement du texte source
- "make": mise à jour automatique des fichiers
  - graphe de dépendance entre des fichiers
  - dates de dernière mise à jour conformes au graphe
  - exécution automatique de commandes si ce n'est pas le cas
- Beaucoup d'outils pour des activités automatisables spécifiques