# Atelier gestionnaires de paquets - Spack

# Commandes de base

## Aide

- `spack help` : donne une description des commandes les plus couramment utilisées
- `spack help <commande>` : donne une description pour une commande particulière

## Lister des paquets

- `spack list <chaine de caractères>` : donne la liste des produits disponibles à l'installation. Si un argument est donné, les paquets ayant la chaine de caractères sont affichés
- `spack find <spec>` : donne la liste des produits installés. Si un argument est donné, les pa
- `spack compilers` : donne la liste des compilateurs utilisables pour construire des paquets

## Information sur les paquets

- `spack info <paquet>` : donne plusieurs informations sur un paquet
- `spack versions <paquet>` : donne les versions connues pour un paquet
- `spack providers <dep-virtuelle>` : donne les paquets correspondants à une dépendance virtuelle

## Création de paquet

- `spack create <url>` : crée une nouvelle recette
- `spack edit <paquet>` : édite la recette d'un paquet

## Installation de paquets

- `spack spec <spec>` : donne la spécification complète d'une instance d'installation pour un paquet.
- `spack install <spec>` : installe un paquet et ses dépendances
- `spack uninstall <spec>` : désinstalle un paquet

# Installation

```
git clone --depth=2 --branch=releases/v1.1 https://github.com/spack/spack.git
source spack/share/spack/setup-env.sh
```

# Travaux pratiques

1. Combien de paquets sont disponibles dans Spack ?
2. Combien de paquets ayant py- dans leur nom sont disponibles ?
3. Quelles sont les versions “sûres” disponibles pour le paquet vim ?
4. Quels paquets peuvent être utilisés pour une dépendance à scalapack ?
5. Quels compilers peuvent être utilisés sur votre machine (suivez les conseils de Spack pour en ajouter le cas échéant) ?

## TP installation basique

Les objectifs de ce TP sont :

- apprendre les commandes de base pour l'installation de paquets
- comprendre la syntaxe utilisée pour les spécifications
- être capable d'installer un paquet en spécifiant :
  - une version particulière
  - des dépendances
  - des variantes

### Connaissances nécessaires

- Sous-commandes
  - list [“spec”]
  - versions “spec”
  - find [“spec”]
  - spec “spec”
  - install “spec”
- Syntaxe des spécifications
  - `@` : numéro de version d'un produit
  - `%` : compilateur à utiliser
  - `^` : dépendance du produit
  - Variantes :
    - `+` : Active une variante booléenne
    - `~` : Désactive une variante booléenne
    - `name=value` : Active une variante non booléenne

## Protocole

### Installation d'un paquet sans dépendance : zlib

1. Vérifier que le paquet est disponible dans Spack
2. Vérifier les versions disponibles
3. Quelles sont les variantes disponibles pour ce paquet ?
   - Quelle est leur valeur par défaut ?
4. Vérifier la concrétisation du paquet. Quel compilateur est utilisé ?
5. Installer la version par défaut du paquet. Quelle version a été installée ?

6. Installer la version sûre la plus ancienne disponible dans Spack
7. Installer une version avec un autre compilateur que celui qui a déjà été utilisé (si possible).
8. Installer une version avec la variante `optimize` modifiée par rapport au défaut.
9. Faire la liste des versions installées pour le paquet.
10. Désintaller la version avec la variante `optimize` modifiée.

#### Installation d'un produit avec dépendance : zip

1. Vérifier que le paquet est disponible dans Spack
2. Quelles sont les versions disponibles ?
3. Vérifier la concrétisation du paquet. Quel compilateur est utilisé ?
4. Installer la version par défaut du paquet. Quelle version a été installée ?
5. D'autres produits ont-ils été installés ? Si oui (divulgâchage : La réponse doit être oui), lesquels ?

## TP modification de paquet existant

Dans ce TP nous allons plonger au coeur des recettes Spack.

Les objectifs sont les suivants : - Comprendre de façon basique comment fonctionne une recette Spack - Être capable de faire des modifications basiques dans les recettes - Savoir ajouter une version - Savoir ajouter une variante

#### Connaissances nécessaires

- Générales
  - Réaliser une empreinte sha256 avec la commande unix sha256sum : `sha256sum fichier`
- Sous-commandes
  - edit

#### Protocole

Le paquet qui va nous servir d'exemple est arpack-ng. Le site du projet est https://github.com/opencollab/arpack-ng.

Une version "dégradée" de la recette est nécessaire pour réaliser cet exercice :

```
wget -O $(spack repo list | awk '{ print $4 }')/packages/arpack_ng/package.py \
  https://sdrive.cnrs.fr/s/qwRsgL3HM26pKJE/download/package.py
```

#### Ajout d'une version

La version 3.9.1 disponible sur le dépôt Github n'est pas connue dans la recette téléchargée.

1. Ajouter cette version dans le fichier de recette. N'oubliez pas l'empreinte !
2. Vérifier que la version est disponible
3. Installer la version 3.9.1 sans activer MPI et en utilisant `autotools` à la place de `cmake` (indice : consulter les variantes disponibles)
4. Est-ce que tout se passe bien ? Hum sans doute que non.
5. Vérifier les log de construction du paquet et corriger l'erreur dans la recette.
6. Essayez d'installer la version 3.8.0 avec `@3.8.0` puis `@=3.8.0`, que se passe-t-il ?

#### Ajout d'une variante

Nous allons maintenant créer une variante pour ce paquet. Par défaut, dans la compilation utilisant CMake, l'installation des exemples est activée (-DEXAMPLES=ON). Cette partie du TP consiste à rendre optionnelle cette installation et à pouvoir l'activer au moment de l'utilisation de spack install.

1. Réaliser la modification pour que la variante soit activable lors de l'utilisation de spack install
2. Faire en sorte qu'elle ait un effet lors de l'installation !!
3. Installer les deux variantes du paquet

## Installation d'un paquet avec une dépendance sur une version de produit déjà installée

Il est parfois utile de s'assurer que Spack va utiliser la version spécifique que vous souhaitez comme dépendance d'une installation. Pour faire cela, il faut indiquer le hash (nous conseillons d'utiliser la partie du hash donnée par `spack find -vl <dependance>`) précédé d'un `/`:

```
spack install <paquet> ^/xgrgvnc
```

#### Exercice

1. Installer un paquet en spécifiant explicitement l'usage d'une dépendence déjà installée via son empreinte.

## Création d'une recette

#### Connaissances nécessaires

- Sous-commandes
  - create

### Protocole

Le paquet qui va nous servir d'exemple est "idrmem", disponible à l'adresse https://sdrive.cnrs.fr/s/G8MCadR7miQjxSA/download/IdrMem_1.4.tar.gz.

### Exercice

1. Utilisez la commande `create` pour créer un squelette pour "idrmem".
2. Testez si le paquet compile.
3. Faites en sortie que le paquet compile correctement.