

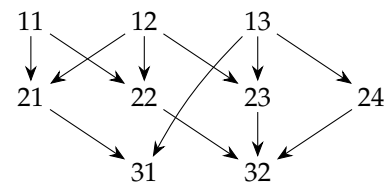
## TP 2 – Mutex et sémaphores

### Exercice 1 – mutex

1. Écrivez un programme composé de 4 nouveaux threads. Chaque thread incrémente un million de fois une variable globale `compteur` de type `long int`. Une fois les threads terminés, affichez la valeur du compteur. Expliquez pourquoi vous n'obtenez pas 4 millions.
2. Pour avoir un résultat conforme aux attentes, on propose d'utiliser une synchronisation à base de *mutex*. Implémentez cette solution.
3. Comparez les performances de la solution correcte avec votre premier programme.

### Exercice 2 – sémaphores et dépendances

Un graphe de dépendances est un graphe orienté dont les sommets sont des « tâches » et les arcs représentent des contraintes d'ordre entre les tâches. La seule contrainte sur le graphe est qu'il soit sans cycle. Nous savons que l'on peut utiliser un sémaphore pour signaler un événement quelconque d'un thread à un autre.



Vous trouverez ci-contre un tel graphe, où l'on voit par exemple que les tâches 11 et 12 doivent être effectuées pour que les tâches 21 et 22 puissent démarrer (et s'exécuter en parallèle). La tâche 32 ne pourra s'exécuter qu'après la fin des tâches 22, 23 et 24.

Écrivez un programme exécutant les tâches de ce graphe en respectant les dépendances. On décide d'allouer un thread à chaque tâche : le code de chaque tâche est un appel à une fonction `tache()` unique, avec en argument le numéro de la tâche (cette fonction se contentera d'afficher le numéro de la tâche.)