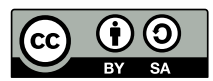
Systèmes d'exploitation et Programmation système (RS)

Lucas Nussbaum <lucas.nussbaum@loria.fr>

Supports de cours, TD et TP largement basés sur ceux de
Martin Quinson <martin.quinson@irisa.fr>

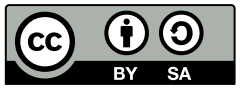
Telecom Nancy – 2ième année

`http://members.loria.fr/lnussbaum/rs.html`



À propos de ce document

Document diffusé selon les termes de la licence



Licence Creative Commons version 3.0 France (ou ultérieure)

Attribution ; Partage dans les mêmes conditions

`http://creativecommons.org/licenses/by-sa/3.0/fr/`

Remerciements

- Sacha Krakoviack pour son cours et Bryant et O'Hallaron pour leur livre
- Les (autres) emprunts sont indiqués dans le corps du document

Aspects techniques

- Document LaTeX (classe latex-beamer), compilé avec latex-make
- Schémas : Beaucoup de xfig, un peu de inkscape

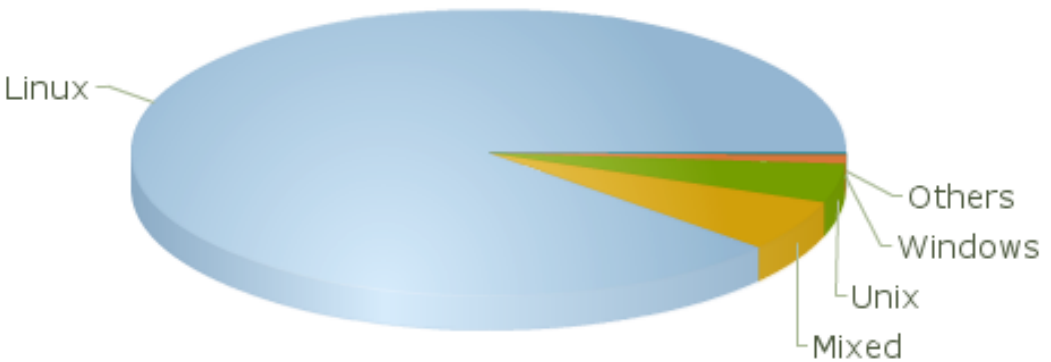
Site du cours : `http://members.loria.fr/lnussbaum/rs.html`

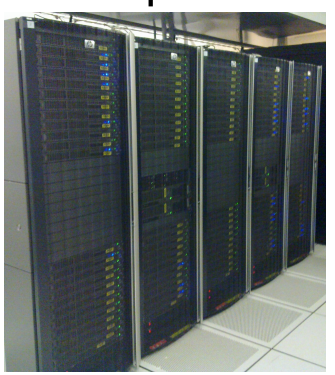
- TD/TP, exams et projets (sources disponibles aux enseignants sur demande)

À propos de moi...

Lucas Nussbaum

- Formation : ingénieur ENSIMAG (2005), Doctorat (2008)
- Depuis 2009 :
  - Enseignant-chercheur (Maître de conférences) à l'univ. de Lorraine
    - Principalement en licence professionnelle ASRALL (Administration de Systèmes, Réseaux et Applications à base de Logiciel Libre)
  - Chercheur dans l'équipe RESIST du LORIA
- Recherche : Systèmes distribués, calcul à haute performance, Cloud





- Contributeur au logiciel libre
  Debian (Project Leader 2013-2015, *Quality Assurance*), Ruby
- Plus d'infos :
  - `http://members.loria.fr/lnussbaum/` (`Lucas.Nussbaum@loria.fr`)

Organisation pratique du module

Module en deux parties

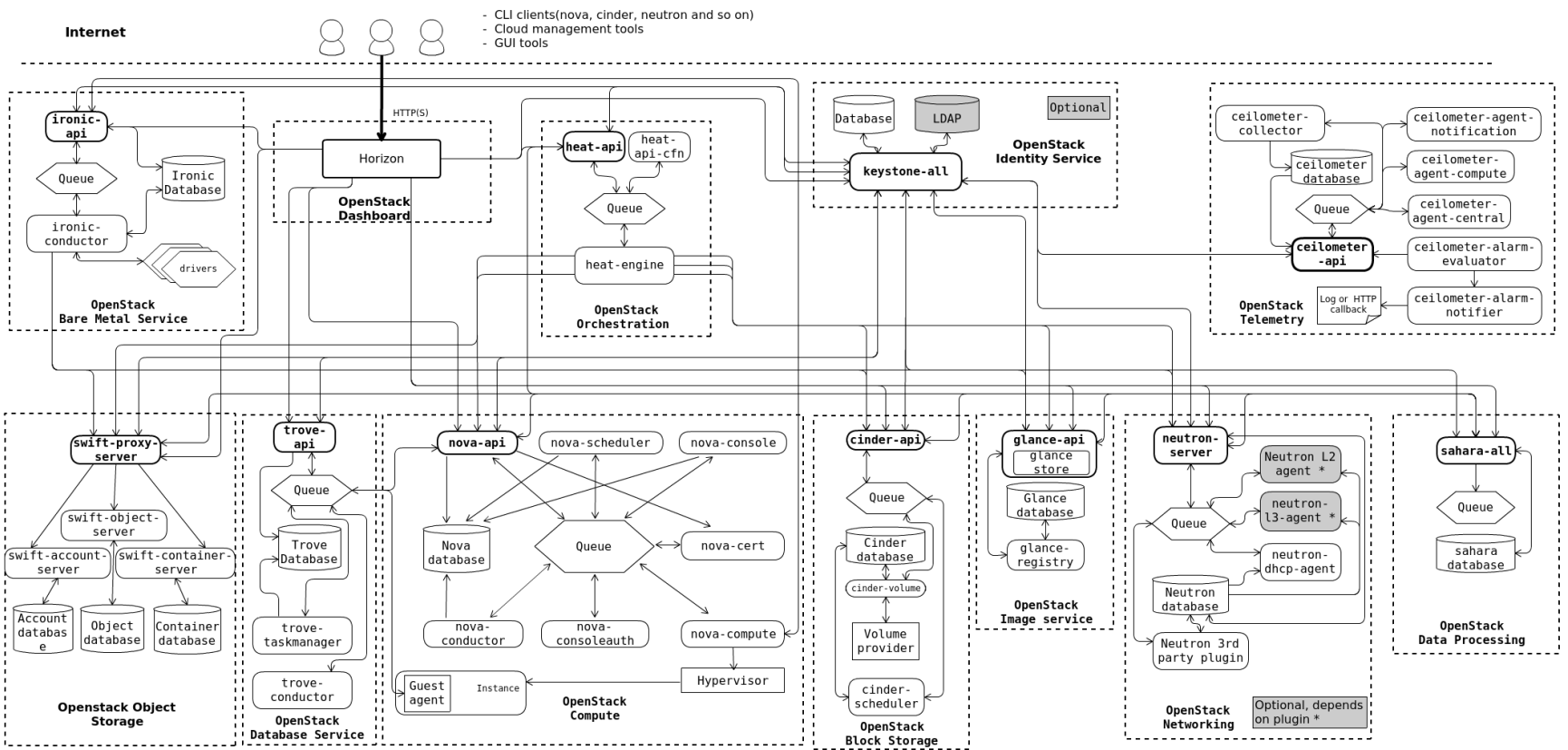
- Partie système (intervenant en cours : Lucas Nussbaum)
  - 5 cours, 3 TD, 3 TP
  - Examen sur table (mi octobre)
    - Documents interdits sauf un A4 recto/verso **manuscrit**
  - un projet (pour décembre)
    - Binômes et Git obligatoires
    - Le sujet arrive bientôt...
- Partie réseaux (intervenant en cours : Isabelle Chrisment)

Implication

- Manipulation : programmez ! Expérimentez !
- Questions bienvenues : pendant/après le cours, par mail, etc.

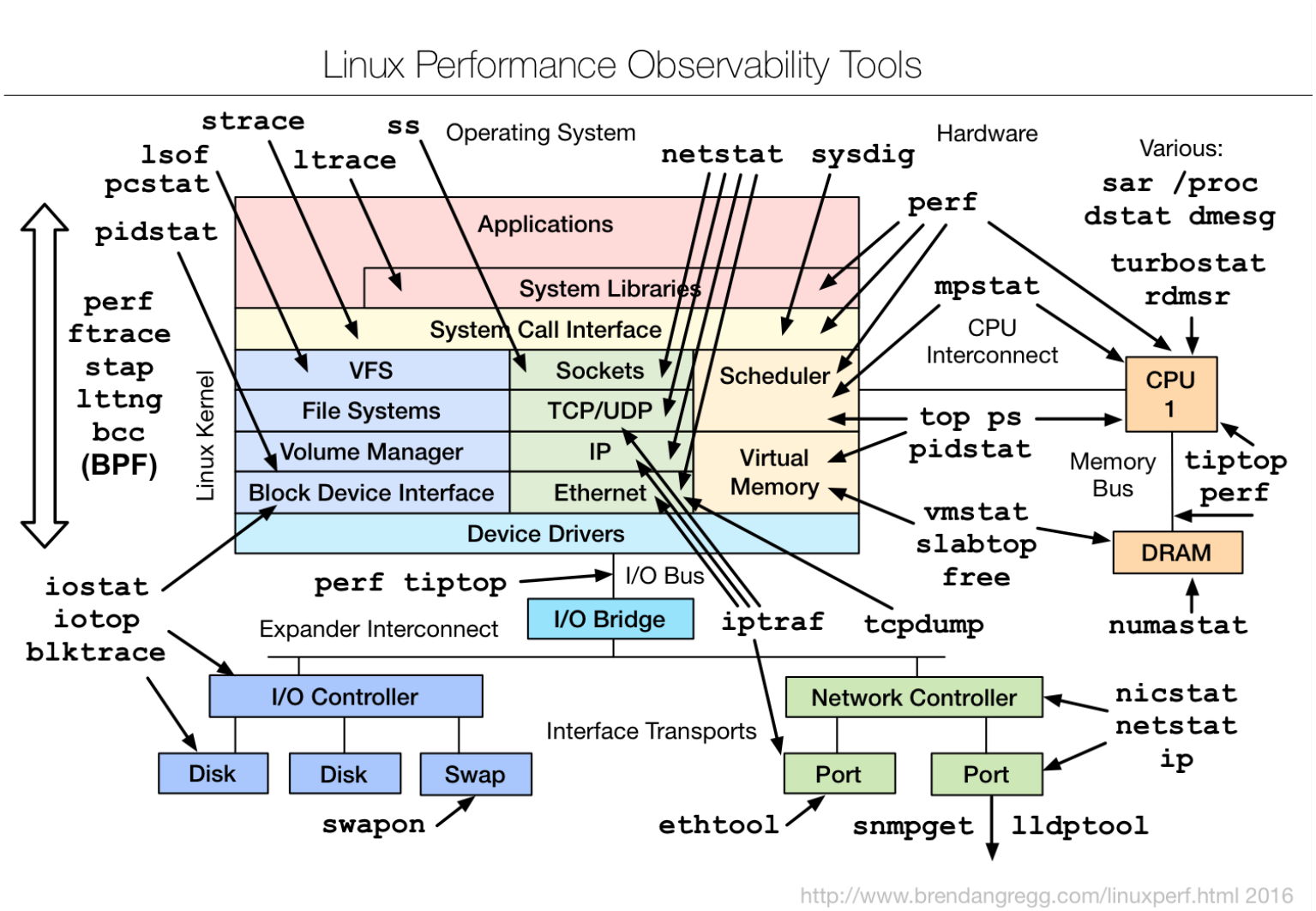
Pourquoi un cours de système ?

- Quatre concepts fondamentaux de l'Informatique (G. Dowek) : Information, **Machine**, Algorithme, Langage
- Les architectures et infrastructures modernes sont complexes
  - Wikipedia : 1145 serveurs, 30 900 CPUs
  - OVH : 250 000 serveurs
  - OpenStack (pile logicielle permettant de créer un *Cloud* privé)

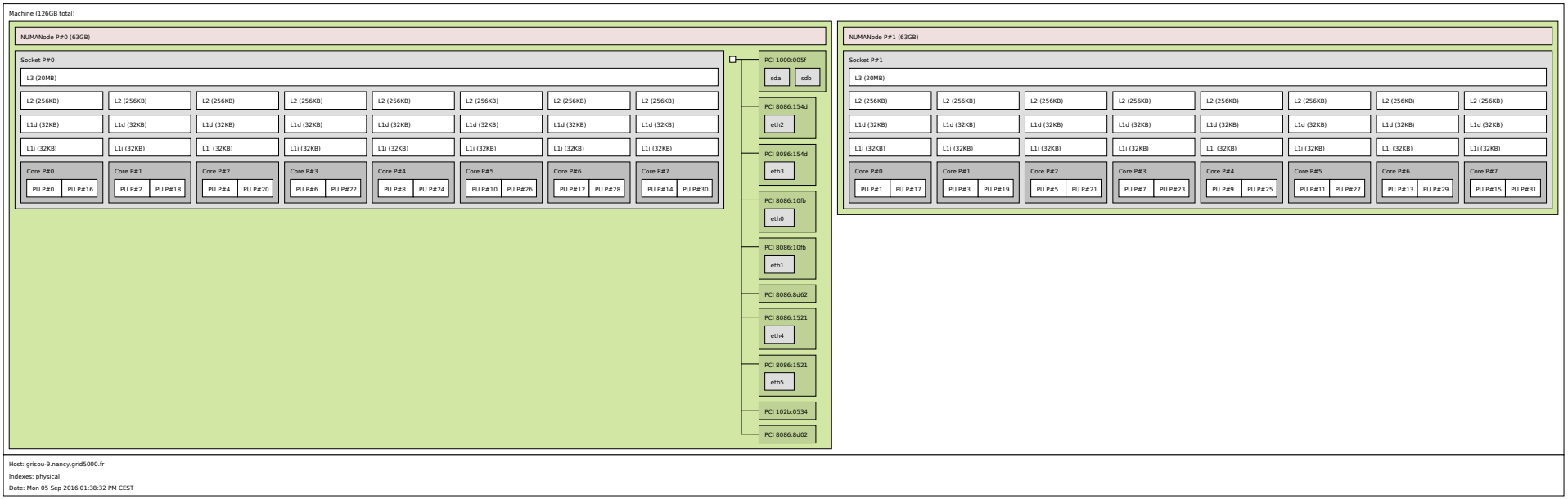


Pourquoi un cours de système ? (2)

- Noyau Linux (et outils d'observation)



Pourquoi un cours de système ? (3)



- Système dual-socket Intel récent (2x Intel E5-2630v3, 8 cœurs/CPU)
  - Hiérarchie de caches entre les processeurs et la mémoire
    - Partagés (L3) ou non (L1, L2) entre cœurs physiques
  - Plusieurs files d'attente pour le même cœur physique (*Hyperthreading*)
  - Mémoire séparée en deux, chaque moitié reliée à un processeur différent
    - Architecture NUMA : Non-Uniform Memory Access
  - Périphériques PCI reliés à un seul des deux processeurs
- Pour l'exploiter pleinement, il faut en comprendre les détails

Pourquoi un cours de système ? (4)

Comment les utiliser efficacement ? Comment les concevoir ?
$\rightsquigarrow$ Performances, sécurité, résilience, efficacité énergétique

Métiers (à la sortie de TELECOM Nancy) :

- **IT Operations / Administration système et réseaux**
  $\rightsquigarrow$ Assurer le maintien en conditions opérationnelles d'une infrastructure (suivi des incidents, montées de version, etc.)
- Mouvement vers le modèle **DevOps** ($\approx$Google Site Reliability Engineers)
  - Suppression des silos *software development* vs *operations*
  - Infrastructure as Code : cloud, *pet vs cattle*
  - Itérations rapides, tests automatiques, déploiement automatiques et continus

  Compétences nécessaires : développement logiciel, compréhension profonde des systèmes (combinaison très recherchée sur le marché du travail)
- **Systèmes embarqués / enfouis** : domotique, automobile, *appliances*
- **Sécurité informatique** (souvent lié à des aspects système/réseau)
- Même comme pur développeur, savoir ce qui se passe sous le capot est utile !

Pourquoi un cours de système ? (5)

Extrait de *The Mythical Man-Month*, Frederick P. Brooks, Jr. (1975) :

Why is programming fun ?

What delights may its practitioner expect as his reward ?

First is the sheer joy of making things. As the child delights in his mud pie, so the adult enjoys building things, especially things of his own design. I think this delight must be an image of God's delight in making things, a delight shown in the distinctiveness of each leaf and each snowflake.

Second is the pleasure of making things that are useful to other people.Deep within, we want others to use our work and to find it helpful. In this respect the programming system is not essentially different from the child's first clay pencil holder "for Daddy's office."

Third is the fascination of fashioning complex puzzle-like objects of interlocking moving parts and watching them work in subtle cycles, playing out the consequences of principles built in from the beginning. The programmed computer has all the fascination of the pinball machine or the jukebox mechanism, carried to the ultimate.

Fourth is the joy of always learning, which springs from the nonrepeating nature of the task. In one way or another the problem is ever new, and its solver learns something : sometimes practical, sometimes theoretical, and sometimes both.

Finally, there is the delight of working in such a tractable medium. **The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures. (...) Yet the program construct, unlike the poet's words, is real in the sense that it moves and works, producing visible outputs separately from the construct itself. It prints results, draws pictures, produces sounds, moves arms.** The magic of myth and legend has come true in our time. One types the correct incantation on a keyboard, and a display screen comes to life, showing things that never were nor could be.

Programming then is fun because it gratifies creative longings built deep within us and delights sensibilities we have in common with all men.

Objectif du module

Utiliser efficacement le système d'exploitation

Contenu et objectifs du module

- Grandes lignes du fonctionnement d'un système d'exploitation (OS)
  Focus sur UNIX (et Linux) par défaut, mais généralisations
- Concepts clés des OS : processus, fichier, édition de liens, synchronisation
- Utilisation des interfaces système : programmation pratique, interface POSIX
- Programmation système (et non programmation interne *du* système)
  Plutôt du point de vue de l'utilisateur (conception d'OS en RSA)

Motivations

- OS = systèmes complexes les plus courants ; Concepts et abstractions claires
- Impossible de faire un programme efficace sans comprendre l'OS
- Comprendre ceci aide à comprendre les abstractions supérieures

Prérequis : Pratique du langage C et du shell UNIX

Bibliographie succincte (pour cette partie)

Livres

- Bryant, O'Hallaron : Computer Systems, A Programmer's Perspective.

Autres cours disponibles sur Internet

- **Introduction aux systèmes et aux réseaux (S. Krakowiak, Grenoble)**
  Source de nombreux transparents présentés ici.
  `http://sardes.inrialpes.fr/~krakowia/Enseignement/L3/SR-L3.html/`
- **Programmation des systèmes** (Ph. Marquet, Lille)
  `http://www.lifl.fr/~marquet/cnl/pds/`
- Operating Systems and System Programming (B. Pfaff, Stanford)
  `http://cs140.stanford.edu/`

Sites d'information

- `http://systeme.developpez.com/cours/`
  Index de cours et tutoriels sur les systèmes

URL du cours : `http://members.loria.fr/lnussbaum/rs.html`

Plan de cette partie du module :

Systèmes d'exploitation et programmation système

1. Introduction
   Système d'exploitation : interface du matériel et gestionnaire des ressources.
2. Processus
   Processus et programme ; Utilisation des processus UNIX et Réalisation ; Signaux.
3. Fichiers et entrées/sorties
   Fichiers et systèmes de fichiers ; Utilisation ; Réalisation.
4. Exécution des programmes
   Schémas d'exécution : interprétation (shell) et compilation (liaison et bibliothèques)
5. Synchronisation entre processus
   Problèmes classiques (compétition, interblocage, famine) ; Schémas classiques.
6. Programmation concurrente
   Qu'est ce qu'un thread ; Modèles d'implémentation ; POSIX threads.