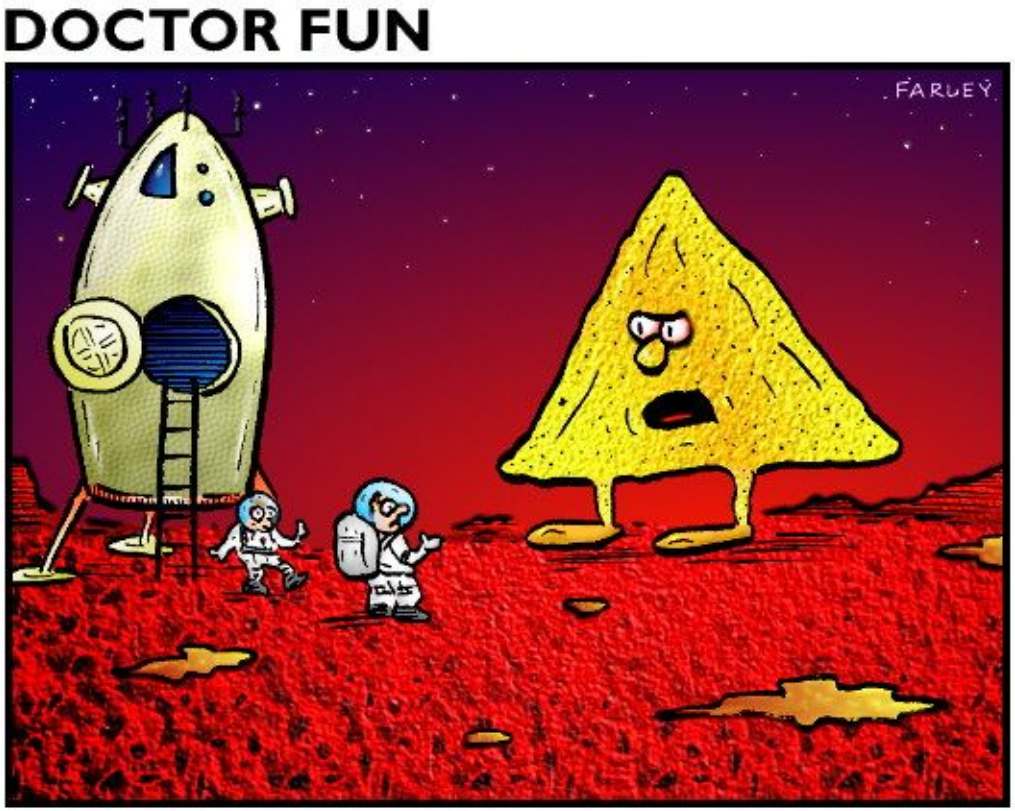


"This is the planet where nachos rule."

- **Etape 3 et 4**
  - Parties mises en commun
  - Toutes parties (inclus Bonus) traitees
- **Etape 5: Gestion disque**
  - Nicolas Mobilia
  - Romain Fornara
  - Jusqu'à partie 6 incluse
- **Etape 6: Network**
  - Barraquand Rémi
  - Savas Ali Tokmen
  - Jusqu'à partie 4 incluse

Etape 3: Threads utilisateurs

- **Classe Thread**
- **Deux types de threads:**
  - Thread kernel: execution de code noyau
  - Thread utilisateur: execution de code MIPS
- **Choix d'implementation**
  - 1 thread kernel <=> 1 thread utilisateur

- **Creation**
  - new Thread(..), Fork(..), InitKernelThread(..), StackAllocate(..), ...
  - MachineState :
    - PCState, StartupPCState, InitialPCState, InitialArgState et WhenDonePCState
  - Stack i386
- **Fonctionnement et limites**
  - Utilisation de scheduler
  - Pas de preemption: non terminaison...
  - Possibilité de debordement du stack i386

- **Classe Scheduler:**
  - Liste de threads en attente
  - Methodes
- **ReadyToRun**
- **FindNextToRun**
- **Run**
  - Sauvegarde registres user + addrspace
  - Check pour overflow
  - Changement de contexte i386: SWITCH
  - Destruction potentielle d'ancien threads
  - Chargement registres user + addrspace + exec

- **Deux type de processus**
  - Processus leger : threads
  - Processus lourd : processus
- **Architecture**
  - Addrspace
    - Pagetable
  - UserRegister
  - Lien de parenté
- **Execution**
  - Machine
  - Préemption

NachOS – Threads utilisateur: processus legers

- **Creation**
  - do_CreateUserThread
    - Addrspace
      - Pointeur vers le parent
    - Stack
      - Partagé avec le parent: probleme d'allocation?
    - UserRegister
      - PCReg, NextPCReg, R4=arg, R31=-1
  - StartUserThread
    - machine->Run()
- **Terminaison**
  - UserThreadExit, pourquoi pas automatique ?

Etape 4: Plusieurs processus

- **Table des pages**
  - Numeros de page virtuelle et de page physique
  - Page valide, page en lecture seule
  - Machine::PageTable et Machine::Translate
- **Fournisseur de frames**
  - Objet global, dans machine
  - Pages occupees et libres
  - Allocation, liberation de pages
  - Mise a 0

- **Elements**
  - MainMemory
  - Registres
  - TLB et PageTable
    - Translate, ReadMem, WriteMem
- **mipssim.cc**
  - Interprete code MIPS
- **Machine::Run**
  - Rappelle le scheduler apres chaque pas: preemptif !

- **Architecture**
  - Similaire aux threads utilisateur
  - Addrspace: nouveau
    - Pagetable
    - Nouveau code: astuce du machine push et pop
  - UserRegister, avec machine initiale
  - Lien de parenté

- **Creation**
  - do_CreateUserProcess
    - Addrspace
      - Contient la table des pages du nouveau processus
    - Stack
      - Un nouveau stack
    - UserRegister
      - Pas d'initialisation
  - StartUserProcess
    - Initialisation des registres
      - Machine initiale: PCReg = 0 , NextPCReg = 4
    - machine->Run()

- **Terminaison**
  - Attente de tous les fils
  - Appel SC_Halt ou SC_Exit:
    - interrupt->Halt() si main
    - Signaler « terminaison » sinon
  - Destruction de l'espace d'adressage
    - Desallouer les frames !

- **Shell**
  - Shell en MIPS !!
  - Commandes classiques: ls, cd, pwd, exit, ...
  - Lancement d'executables UNIX: !<nom>
  - Lancement d'executables NachOS
- **Consommation memoire**
  - Surcharge de new et delete
- **Semaphores utilisateur**
  - Un semaphore utilisateur <=> un semaphore noyau
- **Malloc et free**
  - Allocation et desallocation de page

Etape 5: Fichiers

- **Initialement**
  - 32 pistes
  - 32 secteurs (1 secteur = 128 octets)
  - Espace de stockage total : 128 Ko
  - FreeMapFile
  - directoryFile
- **Modifications**
  - 64 pistes
  - 64 secteurs (taille de secteur inchangée)
  - Espace de stockage total : 512 Ko

- **Répertoire**
  - Table de 10 entrées
- **Entrée de répertoire composée de 3 champs**
  - Un booléen inUse
  - Un entier sector
  - Une chaine de caractère name
- **Convention initiale**
  - Entrée libre : booléen inUse à faux
  - Entrée utilisée : booléen inUse à vrai

- **Modification entrée**
  - Booléen inUse -> isFile
- **Nouvelle convention**
  - Entrée libre : champ sector à -1
  - IsFile à vrai ssi entrée de fichier
  - IsFile à faux ssi entrée de répertoire

- **Réservation des deux premières entrées**
  - Entrée 0 : répertoire « . »
    - Champ sector égal au secteur du répertoire courant
  - Entrée 1 : répertoire « .. »
    - Champ sector égal au secteur du répertoire père
- **Autres entrées pour fichiers et répertoires**
- **Classe FileSystem**
  - CreateDirectory
  - RemoveDirectory
  - ChangeDirectory
  - PrintDirectory

NachOS – Table des fichiers ouverts

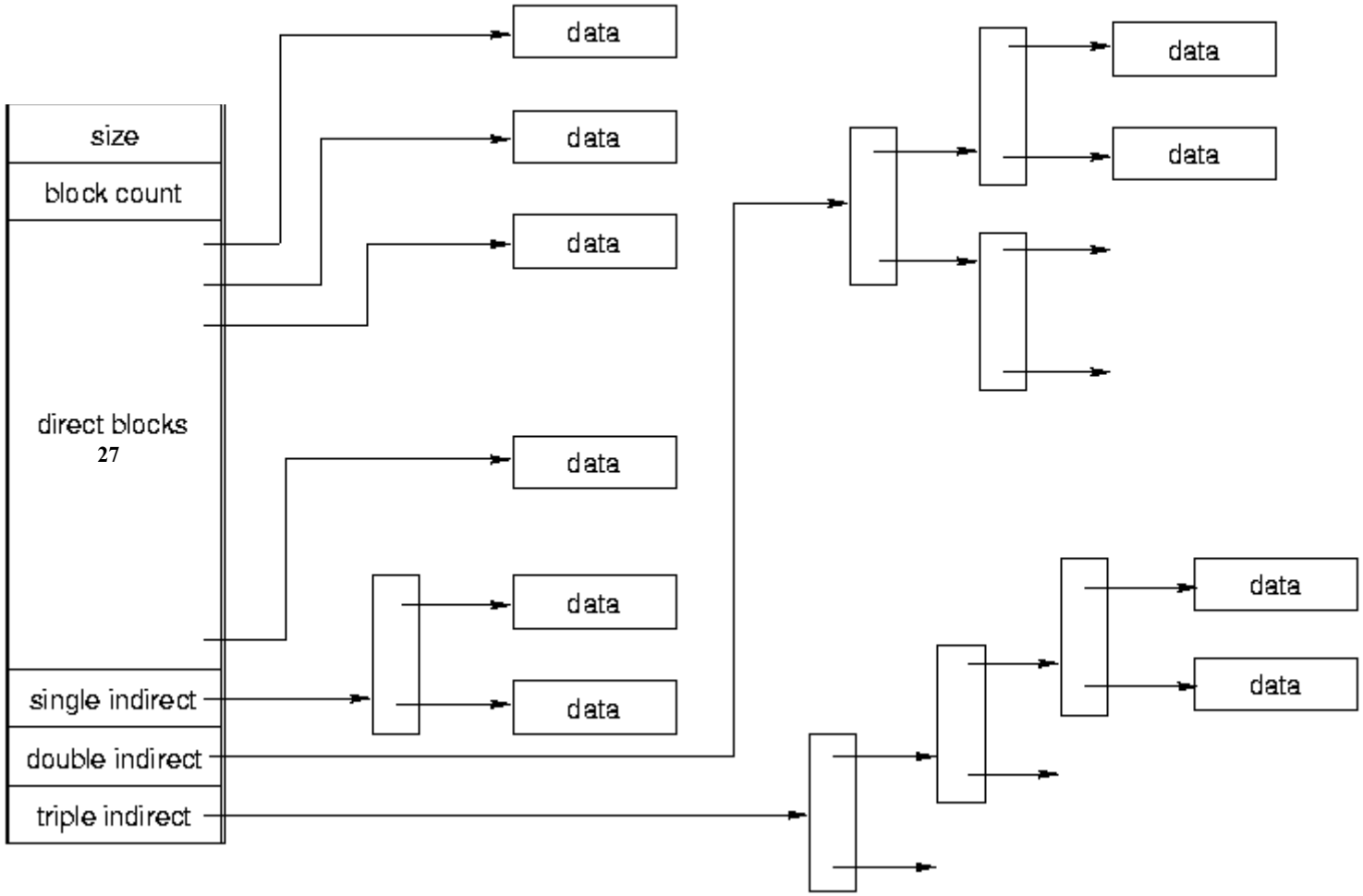
- **Liste**
  - Contient les numéros de secteur des entêtes des fichiers ouverts
  - Sans limite de taille
- **Ajout dans constructeur OpenFile**
- **Suppression dans le destructeur OpenFile**
- **Destructeur FileSystem**
  - Vérification de la liste vide
- **Interdit la double ouverture de fichiers**

- **Ouverture unique d'un fichier**
- **Interdiction de la multi-ouverture de fichiers**
  - Au niveau thread : table des fichiers ouverts
  - Vérification avec la table des fichiers ouverts

- **FreeMapFile et directoryFile jamais libérés**
- **Création d'un destructeur FileSystem**
  - Libération des deux fichiers
  - Libération des autres structures allouées pour le SGF

- **Création dans un entête**
  - 27 secteurs accessibles sans indirection
  - 1 indirection simple
  - 1 indirection double
  - 1 indirection triple
- **Permet d'avoir jusqu'à 4,13 Mo**
- **Classe Indirect**
  - Ne contient que des entrées
  - instance = taille d'un secteur
- **Modification de la classe FileHdr**

NachOS – Schéma indirection



Source : http://istanbulx.acikkod.org/belgeler/gelkitabi/indexnode.html

- **Modification taille du fichier**
  - A la création du fichier
  - Lors de la lecture/écriture du fichier
- **Classe FileHdr**
  - Modification Allocate
    - Alloue plus de mémoire
  - DeallocateFrom
    - Désalloue à partir d'un offset donné

- **Parseur**
  - Automate
  - Découpe le nom de chemin en paramètre
  - Morceaux analysés par l'automate
    - Si correct : action pas à pas
    - Si incorrect : retour en arrière et fin

- **Commande pwd**
  - Chaine de caractère dans l'automate : stringPath
  - Répertoire racine : « / »
  - « .. » : suppression dernier morceau de la chaine
  - Autres chaines : ajout en fin de chaine
- **Sauvegarde du path avant l'automate**
- **Restauration après si erreur**
- **Intégré dans le shell**

Etape 6: Reseau

- **Comparaison avec TCP/IP**
  - Couches 1 et 2 mis ensemble
  - Couche 3: Boites postales (PostOffice)
  - Couche 4: L'application
- **TCP ou UDP ?**
  - Messages UDP sur localhost
  - Pas de transmissions fiables
- **Datagram: 64 octets**
- **Pull vs. Push**

- **Signature des paquets**
  - Par l'expediteur, int
  - int++ a chaque envoi
- **Deux types de paquets**
  - Normal: donnees
  - ACK: accuse de reception

- **Envois asynchrones**
  - SendReliable ajoute a la liste
  - Liste d'attente par boite en sortie
  - Thread envoi en permanence
- **Receptions filtrees**
  - L'utilisateur ne voit pas les ACK
  - L'utilisateur ne voit pas les retransmissons
  - L'utilisateur ne recoit aucun doublon

- **BigSend**
  - Connexion: 2-way handshake
    - Envoi de la taille
    - Utilise SendReliable
  - Decoupe & SendReliable avec petits paquets
  - Attend pour la fin
    - Reponse « reception OK »
    - 1-way drop

- **BigReceive**
  - Attente infinie
    - Attend pour taille
    - Envoie message « tu peux envoyer »
  - Boucle tant que taille recu < taille a recevoir
  - Termine
    - Message « reception OK »
    - Retourne le paquet en char*
      - ATTENTION: ne pas oublier de desallouer !

- **Utilise BigSend et BigReceive**
  - Serveur attend: BigReceive sur la boite 0
  - Client se connecte et envoie: BigSend
  - Quand fichier recu, serveur refait BigReceive
- **Le nom du fichier = premiers octets**
  - Format ASCII char*
  - Peut utiliser . ou / ou encore ..
    - ATTENTION: pas de mkdir !
- **Separateur: '\0'**

- **Limite: une seule reception en meme temps**
  - Car utilise la boite 0
  - Pour resoudre, utiliser deux ports:
    - « controle » sur boite 0
    - « donnees » sur la premiere boite disponible
- **Probleme: plantage client**
  - Solution: attente limitee dans boucle de reception donnee dans BigReceive
- **Probleme: trop de paquets**
  - 4 MO = 65536 paquets !
  - Solution: agrandir taille de datagram