Mots clés

Question 1

- programme en cours d'execution
- code du programme, program counter, pile, données
- PID, ppid, priorité, temps, table des fichiers ouverts, état
- nouveua, pret, en cours, terminé et en attente
- scheduler, privilège des processus IO, alterne, sauvegarde et charge contexte
- FCFS, SJF, priorité (age + priorité), round-robin scheduling (trop grand, trop petit), MQS, MQFS (migration de file en file).
- Partage information calcul, IPC, fichiers, tubes, tubes non-nommés (temporarire, stdin, stdout, stderr, père-fils), tubes nommés (permanent, fichiers spéciaux, processus indépendants), mémoire partagée (taille configurable, commun à plusieurs processus)

Question 2

- point de synchronisation (attente), signaux (SE, SIGINT), sémaphores (variable entière partagée, fonctions atomiques p et v)
- section critique (code exécuté début à la fin sans interruption), partage de ressources, variable partagée (while + mise en pause), alternance (while + complexe), fichier (fichier exclusif, lente et waiting), hardware (while + sémaphores), sémaphores (recommandée, un par processus).
- thread = mini-processus, multithread, partage ressource, plusieurs coeurs, plus léger et rapide que création processus.
  - Noyaux (SE) ou utilisateur (librairie externe). many-to-one = utilisateur (rapide mais pas de support, pas mutli-coeurs, bloquante), one-to-one = noyau (meilleur support, très utlisé, lourd pour SE), many-to-many (pool de thread kernel associé à la volée, complexe à implémenter).
  - fork, execl, terminaison (asynchrone vs différée), signaux.
- Interblocages, ressources limitées, exclusion mutuelle, détention et attente (sinon accumulation et famine), impossibilité de requisitionner, attente circulaire (état sûr, algorithme du banquier, on ne connait pas les besoins)
- Détection, utilisation CPU, waiting. Correctoin : tuer un procesuss, rollback si possible ou ne rien faire.

Question 3

- Ressource indispensable, suite non strucutrée partagée par les processus. Processus en mémoire pour pouvoir s'exécuter. Chaque zone mémoire pour protectoin
- Registres (petite mais rpaide), mémoire vive (plus grande mais moins rapide), mémoire cache (informations fréquemment utilisées), mémoire virtuelle (simulée, disque dur, lente mais abondante)
- adresse logique (programme) ⟶ physique (RAM) par MMU. compilation (plus fait), chargement, exécution (hardware précis).
- Mono programmation (1 processus), toute la mémoire sauf OS vs Multiprogrammation (plusieurs processus)
- Partitions fixes de tailles variables, fragmentation interne
- Partitions variables, table de bits (lent à parcourir, précision des blocs), liste chainée (facile à parcourir et regrouper des blocs, fragmentation interne).
- first-fit (rapide et très utilisé), best-fit (fragmentation externe), worst-fit (fragmentation interne)
- Compactage pour régler fragmentation

- Pagination frames (découpe mem physique), pages (découpe mem logique), table des pages association, mémoire logique contigue, mémoire physique non-contigue. Plus de fragmentation externe, fragmentation interne 1/2 page par processus.
- Segmentation, spéaration types données (données programme, code, etc) pour protection en segments (taille, nom, numéro). Fragmentation externe.
- Segmentation + pagination, ia32, unité segmentation (logique (segment + décalage) ⟶ linéaire (table de page, page, décalage)), unité pagination (linéaire ⟶ physique)

Question 4

- beaucopu programmes, ou programmes > mémoire. mémoire virtuelle via pagination (bit indiquant si page être sur virtuelle) et disque (partition ou fichier)
- si page cherchée non présente (bit 0) ⟶ défaut de page, vérification de l'adressage, frame libre, récupère la page depuis le disque, bit à 1, redémarrage processus
- Si plus de frames dispo on vire une page sur virtuelle, FIFO (page plus ancienne, mauvais), OPT (plus nécessiare avant longtemps, impossible à savoir), LRU (moins récemment utilisé, timestamp, recherche, pile, dérivé),
  - pseudo-LRU, bit référence, 1 lorsqu'utilisé, remis à 0 quand tout utilisé, victime = premier 0. octet référence, seconde chance (bit modification)
  - LFU (mois fréquemment utilisée, mauvais), MFU (palier à LFU mais mauvais)
- Pool de page dispo, et écriture délayée. Allocation frames équitable (beaucoup de gaspillage), allocation proportionnelle (priorité, multiprogrammation), pas assez frames = beaucoup défauts de pages = + de temps à récupérer pages = trashing (mutliprogrammation ⟶ défaut ⟶ waiting ⟶ moins CPU ⟶ multiprogrammation+).
  - Modèle de la localité (identifier pages utilisées ensembles pour minimiser défauts de pages)

Question 5

- collection nommée d'informations relation, séquentiel, direct, indexée. Partitions répertoires, absolu, relatif, liens, montage (implicite et explicite), TFO, TDF, répertoires (liste mais lent, hashmap+liste mais collisions et grandeur varibale)
- allocation contigue (fragmentation externe, buffer = interne + externe), ou éparpillées (ext4), défragmentation dangereuse et chronophage
- Allocation chainée (blocs chainées entre eux, beaucopu voyage tête de lecture donc cluster ⟶ fragmentation interne mais pas externe)
- Allocation indexée (accès direct à certains blocs mais toujours même problème + index)
- Plusieurs allocations différnetes par exemple. Blocs marqués libre (table de bit mais grande, ou liste chainée)
- Vérification cohérence (2 fois libre/occupé/les deux/aucun)
- Sauvegarde (restauration rapide), archivage (garder longtemps). Incrémentale (depuis dernier, rapide, légère mais complexe à restaurer), différentielle (depuis dernière complète, très grand et lent mais simple à restaurer)
- Journal, défaire opérations non terminée, opérations ACID (atomicité, cohérence, isolation, durabilité)

Question 6

- E/S et contrôleurs, interruptions, commandes. Blocs vs caractères. Registres microprocesseur, controlleur. Dialogue via ports E/S (assembleur, registres controleur), registre mappé zone mémoire (simple, opérations et protection de base).
- CPU tout le temps solicité pour transfert donnée en mémoire, DMA controleur spécial pour transfert donnée en mémoire (1 seule interruption à la place de plusieurs) Vol de cycle car ne peut pas attendre par rapport au CPU, un seul accès à la mémoire.

- Interruptions, sauvegarde procesuss en cours, crétaion contexte, exécuter routine, puis redémarre proecssus
- Pilote, chargé au kernel, critique, traduit les informations
- Couche logicielle indépendante (interface standard), uniformiser l'accès, API pilote, buffers, erreurs, allocation et libératoin périphériques non partag, vérifi dispo.
- E/S applicative, libraries systèmes, spooling pour file d'attente des périphériques non partageables

Question 7

- plateaux (contient pistes), cylindre (contient pistes, pos tête de lecture), piste (contient secteurs, plateau + position tête), secteurs (blocs taille fixe). Géométrie variable donc simplifiée. Formatage de bas niveau (écrire secteurs et codes ECC), formattage de haut niveau et partitions (espaces séparés)
- Disk arm schedulling (miniser le seek time total pour plsuieurs requpetes, par SE et controleur). FCFS (simple mais nul), SSTF (plus proches, meilleur mais famine loingtaine), SCAN (d'un bout à l'autre, plus de famine), C-SCAN (SCAN mais que dans un sens), LOOK (SCAN mais jusqu'a requète pas extrémité), C-LOOK (LOOK mais que dans un sens)
- SSTF = meilleur FCFS, C-SCAN, C-LOOK bien pour disques chargés, LOOK ou SSTF souvent utilisé.
- RAID, combiner les disques (mirroring et stripping), 0 = stripping, 1 = mirroring, 3 = stripping octets + parité, 4 = stpirping bloc + partité (plusieurs accès lectures, un seul accès écriture car parité), 5 = RAID 4 + parité distribuée (plusieurs accès lecture, plusieurs accès écriture), 6 = RAID 5 + double-parité distribuée (CPU controleur RAID plus important), RAID 0+1 (stripping dupléque par mirroring, cher). hot-swap et spare. Maximiser débit et diversité.
- Horloge quantum, compteur, registre, etc. Quartz signal périodique, compteur, puis interruption. One-shot (interruption envoyée et attnete), square wave (interruptions répètées, ticks d'horloge). Maintient de la date depuis référence (epoch), quantum, utilisation CPU, alarm, etc. Fil d'évènement pour planifier des évènements à certains ticks.

Question 8

- Protection = protection des ressources des processus. politique de protection = ce qu'il faut protéger, mécanismes protection = comment, objets = ressources matérielles ou logicielles, domaine = ensemble de droits d'accès sur tous les objets. Processus a un domaine, statique (droits restent les même) ou dynamique (droits changent).
- Matrice d'accès, switch, control, owner, copy, transfer. Grande pour implémenter, utilisation d'access list (objet, domaines qui ont des droits dessus). Révocation des droits (quand, pour qui, pour combien de temps, etc)
- protection niveau physique, humain, réseau et système. Informations crutialles, cout de pénétrer système > gain. vol d'info, modification, destruction, utilisation système.
- Identification mdp (keylogger, sniffing, dictionnaire, rainbow, brute force, déduction, fishing). Mdp faibles et réutilisés.
  - en clair = vraiment pas bien, chiffré = dépends de la protection de la clé + même mdp, même cryptogramme, hash = même mdp, même hash ⟶ rainbow table, hash+salt = bien. Hash lent = mieux.
  - Tiers de confiance (OAuth)
- Plusieurs facteurs (chaines hash, TOTP, digipass), codes usages uniques clés hardware (FIDO2, etc).
- Password-less, via hardware (FIDO2), biométrique mais dangereux protection, inchangable, confiedentiel

- Erreurs ⟶ vulnérabilités. Attaques, RCE, SQL, format string, username enum, stack/buffer overflow, DoS, DDoS.
- Programmes malveillant (trojan, backdoor, vers, virus, virus script)
- protection périmètre (firewall réseau), machines (firewall et antivirus, pas d'utilisation d'admin)
- Parc informatique, scanner de vulnerabilité, empreinte des programmes, IDS (intrusion, fail2ban), analyse de fichiers journaux

Question 9

- cacher informations, longueur de clé, niveaux de chiffrements, algorithme (public, sur et indépendant), siècles, substitution, transposition, XOR, masque jetable
- Clé secrète (sym), AES, partage de la clé, grand nombre de clés
- Clé publique (asym), 2 clés, tiers de confiance pour vérifier les clés
- RSA, p = 11, q = 17, e = 7, d = 23, (phi = 160 et n = 187). factorisation de grand nombres, exponentiation modulaire, théorème d'euler, bachet-bezout.
- Signatures cryptographiques (authentique, non falsifiable, non réutilisable, non reniable, non modifiable). Clé secrète = tiers de confiance et deux clés. Clé publique, signer = decrypter, vérifier = chiffrer
- X509 (HTTPS, TLS certificates), tiers de confiance qui lient une origine à une clé privée par une signature. Nav connait clés publiques de tiers de confiances. Let's encrypt moins de vérification. PGP confiance transitive.