Szczegółowe Notatki do Systemów Operacyjnych

Część I: Wprowadzenie do systemów operacyjnych

1. Rodzaje i mechanizmy działania systemu operacyjnego

Co to jest system operacyjny?

To oprogramowanie, które pomaga komputerowi pracować. Możesz go porównać do dyrektora w szkole – mówi, co kto ma robić. System operacyjny kontroluje:

- **Sprzęt:** Takie jak procesor (CPU), pamięć, dyski.
- **Programy:** Uruchamia aplikacje i sprawia, że współdziałają ze sobą.

Rodzaje systemów operacyjnych:

1. **Monolityczny:**
    - **Jak to działa?** Cała "mózgownica" (jądro) systemu zawiera wszystkie narzędzia.
    - **Przykład:** Linux, Unix (w klasycznym podejściu).
    - **Plusy:** Działa szybko, bo wszystko jest razem.
    - **Minusy:** Jeśli coś pójdzie nie tak w jednym miejscu – cały system może mieć problem.
2. **Mikrokernel:**
    - **Jak to działa?** Jądro robi tylko podstawowe rzeczy (zarządza pamięcią, planuje pracę), a wszystko inne działa osobno.
    - **Przykład:** Minix, QNX.
    - **Plusy:** Lepsza izolacja – błąd w jednym programie nie zatrzyma całego systemu.
    - **Minusy:** Może działać trochę wolniej, bo elementy muszą się komunikować.
3. **Hybrid kernel (jądro hybrydowe):**
    - **Jak to działa?** Łączy cechy obu powyższych: trochę wszystkiego jest razem (dla szybkości), a część działa osobno (dla bezpieczeństwa).
    - **Przykład:** Windows NT, macOS.
4. **System warstwowy:**
    - **Jak to działa?** Cały system jest podzielony na "warstwy" (np. sprzęt, sterowniki, aplikacje). Każda warstwa korzysta z usług warstwy poniżej.
    - **Plusy:** Łatwiej znaleźć i naprawić błąd, bo wszystko jest podzielone na mniejsze części.

Mechanizmy działania systemu:

- **Planowanie zadań (scheduler):**
  Dba o to, by każdy program dostał trochę czasu na wykonanie swoich zadań. Porównaj to do rozdawania cukierków dzieciom – każdy dostaje swój kawałek na chwilę.

- **Zarządzanie pamięcią:**
  System przypisuje odpowiednią część pamięci komputerowej (RAM) różnym programom. W skrócie – dba, żeby programy nie zaczęły się wzajemnie "przykładać" i nie krzyczały na siebie o zasoby.

- **Obsługa urządzeń (I/O):**
  Sterowniki to takie programiki, które umożliwiają systemowi "rozmawianie" z urządzeniami (drukarka, klawiatura, monitor).

- **Komunikacja między procesami (IPC):**
  To sposób, dzięki któremu programy wymieniają się informacjami. Przykładowo, jak listonosz przekazuje wiadomości między domami.

Przykładowe pytania:

1. **Pytanie:** Co to jest system operacyjny i co robi?
   Odpowiedź: System operacyjny to program, który zarządza sprzętem komputera i pomaga uruchamiać inne programy. Kontroluje pamięć, procesy, urządzenia wejścia/wyjścia i umożliwia komunikację między aplikacjami.

2. **Pytanie:** Jakie są główne różnice między systemem monolitycznym a mikrokernelowym?
   Odpowiedź: W systemie monolitycznym wszystkie funkcje działają razem w jednym dużym programie (jądrze), co daje szybkość, ale może powodować problemy, jeśli pojawi się błąd. Mikrokernel działa w ten sposób, że tylko podstawowe zadania są wykonywane w jądrze, a reszta działa osobno, co zwiększa niezawodność, ale może powodować spowolnienie przez dodatkową komunikację.

---

2. Zadania poinstalacyjne w systemach operacyjnych

Co to są zadania poinstalacyjne?

To czynności wykonywane po instalacji systemu, żeby wszystko działało sprawnie. Wyobraź sobie, że kupiłaś nowy komputer – musisz najpierw zainstalować aktualizacje, ustawić sieć, zainstalować sterowniki i konta użytkowników.

Najważniejsze zadania:

- **Aktualizacja systemu:**
  Pobieranie najnowszych poprawek i poprawek bezpieczeństwa.
  Przykład w Linux:

```
sudo apt update
sudo apt upgrade
```

- **Konfiguracja sieci:**
  Ustawianie adresu IP, maski, DNS.
  Prosto mówiąc: To jak ustawienie adresu domowego – żeby komputer „wiedział", gdzie jest.

- **Instalacja sterowników:**
  Potrzebne programy, żeby sprzęt (drukarka, karta graficzna) działał poprawnie.

- **Konfiguracja usług:**
  Ustawienie programów takich jak serwer WWW (np. Apache) czy bazy danych.
  Przykład:

```
sudo apt install apache2
```

- **Zarządzanie kontami użytkowników:**
  Tworzenie kont użytkowników i ustawianie, kto co może robić.
  Przykład (Linux):

```
sudo adduser nowy_uzytkownik
sudo usermod -aG grupa nowy_uzytkownik
```

Przykładowe pytania:

1. **Pytanie:** Dlaczego aktualizacje systemu są ważne?
   Odpowiedź: Ponieważ zapewniają, że system jest bezpieczny, działa poprawnie i zawiera najnowsze poprawki do błędów.

2. **Pytanie:** Co zwykle robi się zaraz po zainstalowaniu systemu?
   Odpowiedź: Aktualizuje się system, ustawia konfigurację sieci, instaluje sterowniki oraz dodaje użytkowników i ustawia im odpowiednie uprawnienia.

---

3. Zarządzanie użytkownikami, uprawnienia

Co to jest zarządzanie użytkownikami?

Użytkownik to konto, którym się logujesz. Zarządzanie nim polega na tworzeniu, edytowaniu lub usuwaniu tych kont oraz ustalaniu, co każdy użytkownik może robić.

Jak to działa:

- **W systemie Linux:**
  Informacje o użytkownikach znajdują się w plikach takich jak `/etc/passwd`, `/etc/shadow` (dla haseł) oraz `/etc/group` (dla grup).
  Przykład polecenia:

```
sudo adduser imie
sudo usermod -aG grupa imie
```

- **Uprawnienia:**
  Są oznaczone jako *read* (r), *write* (w) oraz *execute* (x).
  Przykład:
  Jeśli plik ma ustawione uprawnienia **rwx** dla właściciela, oznacza to, że właściciel może czytać, pisać i uruchamiać dany plik.

Przykładowe pytania:

1. **Pytanie:** Gdzie w systemie Linux są przechowywane dane o użytkownikach?
   Odpowiedź: Dane o użytkownikach są w pliku `/etc/passwd` (i hasła w `/etc/shadow`), a grupy w pliku `/etc/group`.

2. **Pytanie:** Co oznacza, że plik ma uprawnienia rwx?
   Odpowiedź: "r" oznacza, że można czytać plik, "w" – że można modyfikować plik, a "x" – że można go wykonywać lub (w przypadku katalogów) wchodzić do nich.

---

4. Instalacja i konfiguracja oprogramowania

Jak instalujemy oprogramowanie?

Wyobraź sobie, że pobierasz i instalujesz aplikację na telefonie. W komputerach najpierw pobieramy program, potem go instalujemy, a potem konfigurujemy tak, aby działał właściwie.

Etapy instalacji:

1. **Pobranie oprogramowania:**
   Pobierasz plik z Internetu lub z repozytorium (pomyśl o tym jak pobieranie aplikacji z App Store).

2. **Instalacja:**
   W systemie Linux używamy narzędzi, które automatycznie instalują program i rozwiązują zależności (np. `apt install nazwa_pakietu`).
   W Windows często uruchamiamy instalator (.exe) i postępujemy zgodnie z instrukcjami.

3. **Konfiguracja:**
   Ustawiasz opcje programu – edytujesz pliki konfiguracyjne lub używasz graficznego interfejsu, żeby wszystko działało tak jak chcesz.

4. **Testowanie:**
   Sprawdzasz, czy program działa poprawnie – czy np. serwer www naprawdę wyświetla stronę.

Schemat instalacji:

```
flowchart TD
    A[Pobranie pakietu] --> B[Sprawdzenie, czy pakiet jest poprawny]
    B --> C[Instalacja pakietu]
    C --> D[Instalacja dodatkowych potrzebnych programów]
    D --> E[Konfiguracja oprogramowania]
    E --> F[Testowanie działania]
```

Przykładowe pytania:

1. **Pytanie:** Jakie są najważniejsze kroki przy instalacji oprogramowania?
   Odpowiedź: Najpierw pobieramy oprogramowanie, potem instalujemy (razem z zależnościami), konfigurujemy ustawienia i w końcu testujemy, czy działa poprawnie.

2. **Pytanie:** W jaki sposób sprawdzamy poprawność pobranego pliku?
   Odpowiedź: Możemy użyć sumy kontrolnej (checksum) lub podpisu cyfrowego, aby upewnić się, że plik nie został zmieniony i pochodzi z zaufanego źródła.

Część II: Systemy operacyjne – bardziej szczegółowe mechanizmy

---

1. Struktury systemów operacyjnych

Jak zbudowany jest system operacyjny?

Możesz to porównać do budynku. Różne podejścia:

1. **Monolityczny:**
   Wszystko (sterowniki, pamięć, procesy) znajduje się w jednym "budynku" (jądrze).
   - **Plus:** Szybko działa, bo wszystko jest blisko.
   - **Minus:** Gdy pojawi się problem w jednej części, cały budynek może mieć awarię.

2. **Mikrokernel:**
   Jądro robi tylko najważniejsze zadania, a reszta to oddzielne pokoje.
   - **Plus:** Gdy jeden pokój ma problem, nie wpływa to na resztę.
   - **Minus:** Więcej "chodzień" między pokojami, co może spowalniać pracę.

3. **Hybrid kernel:**
   Mieszanka – część spraw działa razem, część osobno, żeby wyważyć szybkość z niezawodnością.

4. **Warstwowy:**
   System dzieli się na warstwy (jak piętra w budynku) – od sprzętu, przez sterowniki, aż po aplikacje.
   - **Plus:** Łatwiej znaleźć problem w danej warstwie.

Diagram struktury warstwowej:

```
+--------------------+
| Aplikacje          |
+--------------------+
| Biblioteki         |
+--------------------+
| Warstwa systemowa  |
+--------------------+
| Sterowniki         |
+--------------------+
| Sprzęt             |
+--------------------+
```

Przykładowe pytania:

1. **Pytanie:** Jakie są zalety architektury mikrokernelowej?
   Odpowiedź: Mikrokernel wykonuje tylko podstawowe funkcje, a reszta działa oddzielnie. Dzięki temu, gdy jeden element zawiedzie, nie zawiesi to całego systemu, co zwiększa bezpieczeństwo.

2. **Pytanie:** Co to jest hybrid kernel?
   Odpowiedź: Hybrid kernel to model łączący zalety jądra monolitycznego i mikrokernelowego – część funkcji jest w jądrze (dla szybkości), a część poza nim (dla niezawodności).

---

2. Zarządzanie procesami

Co to jest proces?

Proces to po prostu działający program. Wyobraź sobie, że otwierasz przeglądarkę – to jest proces. System operacyjny dba o:

- **Tworzenie i zamykanie procesów:** Kiedy uruchamiasz lub zamykasz program.
- **Planowanie:** Decyduje, który proces dostanie chwilę pracy na CPU.
- **Przełączanie kontekstu:** System zapisuje stan jednego procesu i wznawia inny (jak pauzowanie filmu i potem jego wznawianie).

Przykładowy kod (C) dla tworzenia procesu:

```c
pid_t pid = fork();
if (pid == 0) {
    // To jest proces potomny, uruchamiamy inny program
    execl("/bin/ls", "ls", (char *)NULL);
} else if (pid > 0) {
    // Proces macierzysty - może czekać na zakończenie potomka
    wait(NULL);
} else {
```

```
    perror("Błąd funkcji fork");
}
```

Diagram cyklu życia procesu:

```
flowchart TD
    A[Proces utworzony] --> B[Gotowy do działania]
    B --> C[Wykonywany na CPU]
    C --> D[Proces blokowany (czeka na coś)]
    D --> B
    C --> E[Zakończenie procesu]
```

Przykładowe pytania:

1. **Pytanie:** Co to jest przełączanie kontekstu?
   Odpowiedź: To proces zapisywania stanu jednego procesu (wszystkich jego zmiennych i rejestrów) i przywracania stanu innego procesu, dzięki czemu można wykonywać wiele zadań w tym samym czasie.
2. **Pytanie:** Czym jest algorytm Round Robin w planowaniu procesów?
   Odpowiedź: Algorytm Round Robin przydziela każdemu procesowi określony, równy czas (kwant czasu) na działanie, po czym przełącza się do kolejnego, żeby każdy miał szansę na pracę.

---

3. Synchronizacja procesów

Co to jest synchronizacja?

Synchronizacja oznacza, że system dba, aby procesy nie "walczyły" o te same zasoby jednocześnie. Wyobraź sobie kilku uczniów chcących korzystać z jednego długopisu – trzeba ustalić zasady, kto i kiedy może go użyć.

Główne problemy:

- **Race condition (wyścig):**
  Kiedy dwa procesy próbują jednocześnie zmienić te same dane bez współpracy – może powstać bałagan.
- **Deadlock (zakleszczenie):**
  Procesy czekają na siebie nawzajem, przez co żaden nie może dokończyć pracy.
- **Starvation:**
  Jeden proces cały czas nie dostaje zasobu, bo inne procesy zawsze mają pierwszeństwo.

Metody synchronizacji:

1. **Mutex:**
   Jest jak zamknięcie drzwi – tylko jeden proces może wejść do „krytycznego pokoju” (sekcji krytycznej)

na raz.
Przykład pseudokodu:

```
mutex_lock(&mutex);
// Tu wykonujemy ważne zadania, które nie mogą być przerywane
mutex_unlock(&mutex);
```

2. **Semafory:**
   Są licznikiem, który pozwala określić, ile procesów może jednocześnie korzystać z zasobu.

3. **Zmienne warunkowe:**
   Umożliwiają procesom czekanie na określony warunek – gdy warunek się spełni, procesy zostają poinformowane, że mogą kontynuować.

Przykładowe pytania:

1. **Pytanie:** Co to jest race condition?
   Odpowiedź: Race condition to sytuacja, gdy dwa lub więcej procesów próbują jednocześnie zmieniać te same dane bez odpowiedniej synchronizacji, co może spowodować błędne wyniki.

2. **Pytanie:** Jak zmienna warunkowa pomaga w synchronizacji?
   Odpowiedź: Zmienna warunkowa pozwala procesowi czekać, aż zostanie spełniony określony warunek, zanim zacznie działać, co zapobiega konfliktom przy dostępie do współdzielonych zasobów.

---

4. Zarządzanie pamięcią operacyjną

Co to jest pamięć operacyjna (RAM)?

To pamięć, w której komputer trzyma dane i programy, które są właśnie używane. RAM jest bardzo szybki, ale wszystko z niego znika po wyłączeniu komputera.

Techniki zarządzania pamięcią:

- **Paginacja:**
  Wyobraź sobie, że pamięć to ogromna książka podzielona na małe strony. System przydziela "strony" pamięci do programów.
  - **Page fault:** Gdy program potrzebuje strony, której nie ma w pamięci, system ładuje ją z dysku.
- **Segmentacja:**
  Pamięć dzielimy na segmenty – jak rozdziały w książce (np. kod, dane, stos).
  - **Plus:** Pozwala przypasować pamięć dokładnie do potrzeb programu.
  - **Minus:** Może powstać problem ze "zmarnowaną" pamięcią, jeśli segmenty są różnej wielkości.

- **Pamięć wirtualna i swapping:**
  Jeśli brakuje fizycznej pamięci (RAM), komputer "pożycza" trochę miejsca z dysku, co nazywamy swapowaniem. To wolniejsze, ale pomaga uruchomić więcej programów jednocześnie.

Przykładowe pytania:

1. **Pytanie:** Czym jest paginacja?
   Odpowiedź: Paginacja to metoda dzielenia pamięci na mniejsze, stałe kawałki zwane stronami. Gdy program potrzebuje danych, a odpowiednia strona nie jest w RAM-ie, system ładuje ją z dysku (page fault).
2. **Pytanie:** Co to jest pamięć wirtualna?
   Odpowiedź: Pamięć wirtualna pozwala komputerowi działać tak, jakby miał więcej pamięci (RAM), niż faktycznie jest, dzięki wykorzystaniu części dysku jako dodatkowej pamięci (swap).

---

5. Zarządzanie pamięcią masową

Co to jest pamięć masowa?

To miejsce, gdzie przechowywane są dane na dłuższy czas – na przykład dysk twardy lub SSD. Jest wolniejsza niż RAM, ale dane nie znikają po wyłączeniu komputera.

Główne zagadnienia:

- **Systemy plików:**
  To sposób organizacji plików na dysku. Przykłady: NTFS (Windows), ext4 (Linux), FAT32.
- **Operacje I/O:**
  Są to operacje odczytu i zapisu. Aby przyspieszyć te operacje, stosuje się **buforowanie** – przechowywanie często używanych danych w szybkiej pamięci podręcznej.
- **Hierarchiczna struktura katalogów:**
  To sposób organizacji plików w foldery i podfoldery, co ułatwia znalezienie potrzebnych danych.

Diagram przykładowej struktury katalogów:

```
/
├── bin
├── etc
├── home
│   ├── alice
│   └── bob
└── var
```

Przykładowe pytania:

1. **Pytanie:** Co to jest system plików?
   Odpowiedź: System plików to sposób organizacji i przechowywania danych na dysku. Zarządza plikami, folderami i metadanymi (takimi jak prawa dostępu, rozmiar pliku).

2. **Pytanie:** Dlaczego stosujemy buforowanie w operacjach na dysku?
   Odpowiedź: Buforowanie pozwala przechowywać często używane dane w szybkiej pamięci, dzięki czemu operacje odczytu i zapisu są szybsze, ponieważ nie musimy ciągle korzystać z wolniejszego dysku.

---

6. System plików

Co to jest system plików?

Wyobraź sobie, że to ogromna biblioteka, w której każdy plik jest książką, a katalogi to regały. System plików dba o:

- **Pliki:** Główne "książki" (dane).
- **Katalogi (foldery):** Miejsca, gdzie te książki są trzymane.
- **Metadane:** Informacje o plikach, np. kto i kiedy je edytował.

Typy systemów plików:

- **ext4:**
  Popularny w systemach Linux, dobrze radzi sobie z zarządzaniem danymi i jest odporny na fragmentację.

- **NTFS:**
  Stosowany w Windows, oferuje dodatkowe funkcje bezpieczeństwa i dziennikowanie zmian (logowanie modyfikacji).

- **FAT32/exFAT:**
  Często stosowane na nośnikach zewnętrznych, bo są kompatybilne z wieloma systemami.

Przykładowe pytania:

1. **Pytanie:** Jakie informacje przechowuje system plików oprócz samych danych?
   Odpowiedź: System plików przechowuje metadane, czyli dane o plikach, np. kto ma do nich dostęp, kiedy były modyfikowane oraz rozmiar pliku.

2. **Pytanie:** Jak hierarchiczna struktura katalogów ułatwia zarządzanie danymi?
   Odpowiedź: Dzięki hierarchicznej strukturze możesz łatwo znaleźć potrzebne pliki, ponieważ są one pogrupowane w foldery i podfoldery według rodzaju, właściciela lub innych kryteriów.

---

7. Bezpieczeństwo i ochrona w systemach operacyjnych

Dlaczego bezpieczeństwo jest ważne?

Bezpieczeństwo w systemie operacyjnym chroni nasze dane i zasoby przed nieautoryzowanym dostępem – to jak solidne zamki i alarmy w domu.

Podstawowe mechanizmy:

- **Kontrola dostępu:**
  Używamy uprawnień (rwx) do określenia, kto może czytać, pisać lub wykonywać dany plik.
  ACL: Pozwalają na bardziej szczegółowe ustawienie uprawnień.
- **Uwierzytelnianie:**
  To sposób, w jaki system sprawdza, kim jesteś – przy użyciu loginu, hasła lub dodatkowych metod (np. token czy SMS).
- **Firewall:**
  To taka straż graniczna systemu – blokuje niepożądany ruch sieciowy.
- **Szyfrowanie:**
  Szyfrujemy dane, aby były nieczytelne dla osób, które nie mają odpowiedniego klucza – nawet gdy ktoś ukradnie dysk, nie odczyta danych.
- **Audyt i logowanie:**
  System zapisuje, co się dzieje, żeby w razie problemu można było sprawdzić, kto co robił.

Przykłady narzędzi:

- **SELinux/AppArmor:**
  Specjalne systemy bezpieczeństwa w Linux.
- **Windows Defender, BitLocker:**
  Narzędzia Windows do ochrony i szyfrowania dysków.

Przykładowe pytania:

1. **Pytanie:** Co to jest kontrola dostępu i jak działa?
   Odpowiedź: Kontrola dostępu to system ustawień, który decyduje, kto może czytać, modyfikować lub uruchamiać pliki. Działa to poprzez uprawnienia ustawione na plikach (r, w, x) oraz bardziej szczegółowe ACL, które precyzyjnie określają prawa poszczególnych użytkowników lub grup.
2. **Pytanie:** Dlaczego szyfrowanie danych zwiększa bezpieczeństwo?
   Odpowiedź: Szyfrowanie koduje dane, dzięki czemu nawet jeśli ktoś uzyska dostęp do dysku, nie odczyta danych bez odpowiedniego klucza deszyfrującego.

Podsumowanie

Te notatki krok po kroku tłumaczą działanie systemów operacyjnych w prosty sposób, wyjaśniając każdy element:

- **Wprowadzenie:** Co to jest system operacyjny, jakie są jego rodzaje i podstawowe funkcje.

- **Zadania poinstalacyjne:** Co robi system zaraz po instalacji – aktualizacje, konfiguracja sieci, instalacja sterowników, ustawianie kont.
- **Zarządzanie użytkownikami:** Jak tworzymy i zarządzamy użytkownikami oraz ich uprawnieniami.
- **Instalacja oprogramowania:** Jak pobieramy, instalujemy i konfigurujemy programy.
- **Zaawansowane mechanizmy:** Struktura systemu, zarządzanie procesami, synchronizacja, zarządzanie pamięcią operacyjną i masową, system plików.
- **Bezpieczeństwo:** Jak chronić system przy użyciu kontroli dostępu, uwierzytelniania, zapór sieciowych i szyfrowania.