Architektúra OS

Komponenty OS

Veľký a zložitý systém, akým je OS sa dá vytvoriť a spravovať, ak pozostáva z menších častí – komponentov s dobre definovaným rozhraním a chovaním. Tieto komponenty je užitočné si rozdeliť podľa poskytovanej funkčnosti.

1. Správa procesov.

Proces

je program (kód), ktorý sa vykonáva, čo v multiprocesovom systéme znamená, že je natiahnutý v operačnej pamäti a raz za čas sa dostáva k CPU (je dokonca možné, aby rovnaký program sa v rovnakom čase vykonával viackrát, t.j. dal vzniknúť niekoľkým procesom).

Procesy majú požiadavky na zdroje systému

(CPU, op. pamäť, súbory, V/V zariadenia), tieto požiadavky môžu oznámiť pri svojom vytvorení (=natiahnutí kódu do op. pamäti) alebo neskôr a môže im byť vyhovené hneď pri vytvorení alebo priebežne neskôr.

Procesy môžu vytvárať iné procesy (potomkov),

môžu sa vykonávať súbežne so svojimi potomkami a dokonca "čakať" na ich výsledky.

Okrem procesov spustených používateľom beží v modernom OS niekoľko **obslužných systémových procesov**.

OS umožní:

- vytváranie a ukončenie procesov,
- pozastavenie a reaktiváciu procesov,
- synchronizáciu procesov vrátane riešenia vzájomného zablokovania,
- komunikáciu medzi procesmi.

2. Správa pamäti.

Op. pamäť je kľúčovým prvkom systému a nikdy ho nie je dosť. Preto:

Procesy súťažia o pamäť

Vykonávaný proces musí mať svoj kód a dáta v operačnej pamäti (presnejšie: práve vykonávaný úsek kódu a práve používané dáta), ale práve iné úseky kódu a dát môžu byť odsunuté do sekundárnej pamäti. Okrem toho novovytvárané procesy čakajú na natiahnutie do pamäti a OS im musí prideliť priestor (alebo odmietnuť ich vytvorenie), bežiace procesy môžu požiadať o ďalšiu pamäť (operátor new v Pascale a C++).

OS zabezpečí:

- prehľad o obsadenosti úsekov pamäti,

- pridelenie pamäti novým procesom, prípadne výber procesu na spustenie s uvážením jeho pamäťových požiadaviek,
- pridelenie a vracanie pamäti procesom.

3. Správa sekundárnej pamäti.

Programy a dáta sú uložené v sekundárnej pamäti (hlavne disky), ktorá musí umožniť on-line (priamy) prístup k nim. Procesy tiež môžu pracovať so súbormi (vstup, výstup, dočasné súbory) uloženými na diskoch. V multiprocesovom systéme môže existovať viacero súbežných V/V požiadaviek na disk a OS ich musí koordinovať.

OS musí:

- spravovať voľný priestor,
- prideľovať priestor na diskoch,
- riadiť prístup k disku

4. Správa V/V zariadení.

OS by mal skrývať hardwarové špecifiká V/V zariadení ("copy sprava.txt lpt1" funguje (skoro) rovnako s ľubovoľnou tlačiarňou).

Správa V/V poskytuje:

- systém bafrovania V/V,
- spoločné rozhranie na ovládače periférií,
- ovládače konkrétnych periférií.

5. Správa súborov.

Súbory (file) sú prirodzenou štruktúraciou dát uložených v počítačovom systéme (v UNIXe dokonca platí slogan "všetko je súbor").

Súbor

je logická jednotka dát, abstrahovaná od fyzického spôsobu uloženia.

Správa súborov teda tvorí koncepčne vrstvu nad správou sekundárnej pamäti. Ďalším dôležitým pojmom je:

Adresár (directory)

umožňuje hierarchické členenie súborov do skupín podľa vlastníka, obsahu, určenia, atď (používatelia majú voľnosť v spôsobe hierarchizácie).

OS zabezpečí:

- vytváranie a mazanie súborov a adresárov,
- ďalšie prirodzené operácie (čítanie, zmena obsahu, vlastností),
- zálohovanie súborov,
- mapovanie na sekundárnu pamäť

("100. až 200. riadok je uložený v 232. sektore disku").

6. Systém zabezpečenia.

Vzájomná ochrana procesov a používateľov podľa princípov minulej kapitoly, ale aj zabezpečenie prístupu na úrovni sekundárnej pamäti, súborov, periférií. Spočíva v autentikácii (heslo) a v definovaní práv prístupu a spôsobe ich kontroly pri prístupe k zdrojom.

7. Komunikačný systém (sieťová podpora)

OS musí zabezpečiť komunikáciu medzi systémami na úrovni zdieľania zdrojov (periférie, ale aj CPU – výpočtové servre), ako

aj komunikáciu medzi používateľmi systémov (e-mail, talk, ...) a interaktívne využitie vzdialených systémov (telnet, www, ...). Na systémovej úrovni musí umožniť aj správu komunikačného systému.

8. Interpret príkazov

Ako poslednú sme nechali komunikáciu používateľa so systémom – zadávanie príkazov a oznamovanie ich výsledkov. Túto komunikáciu zabezpečuje interpret príkazov (command interpreter, shell). Môže prebiehať na úrovni kontrolných príkazov (štítkov) v dávkových systémoch, alebo ako interaktívny program (time-sharing systémy): riadkovo orientovaný alebo grafický. Základný účel je vždy rovnaký: získať ďalší príkaz, vykonať ho a oznámiť, ako to dopadlo.

Čo z toho vidí používateľ?

Používateľ komunikuje so sytémom prostredníctvom interpreta príkazov. Jednotlivé funkcie (zobraziť obsah súboru, ukončiť proces, vytlačiť súbor, pozhovárať sa s kolegom cez talk, ...) má prístupné ako príkazy alebo ikony. Tu sa stiera hranica medzi systémom a aplikačnými programami – väčšinu systémových

programov, ktoré sú mimo jadra vieme nahradiť alternatívnym aplikačným programom.

Čo z toho vidí programátor?

Programátor (systémových alebo aplikačných) programov vidí API (applications programming interface) tvorené volaniami systému, ktoré realizujú funkcie popísané vyššie. Tieto volania môžu byť vo forme prerušení (DOS), alebo viac - menej štandardizovaného rozhrania napr. unixovského typu (POSIX).

Štruktúra OS

Komponenty popísané vyššie sa musia dať dohromady, aby vznikol funkčný OS.

Mnohé systémy nemajú skoro žiadnu štruktúru,

vznikli malé a neštrukturované a postupne sa rozrastali. Jednotlivé komponenty nie sú dobre oddelené a nerešpektujú modularitu (ak používam služby rozhrania, nepotrebujem vedieť, ako je implementované a mal by som mať len obmedzený prístup k vnútornostiam tejto implementácie).

Štruktúra MS-DOSu

pozostáva z vrstiev (ovládače periférií a služby na úrovni BIOSu, DOSovské ovládače a služby, interpret príkazov (command.com) a aplikačné programy), pričom vyššie vrstvy (aj aplikačné programy) majú prístup ku všetkým nižším vrstvám vrátane HW.

Pôvodné verzie UNIXu

boli tiež málo štrukturované. Pozostávajú z ovládačov periférií, jadra (poskytuje kompletnú funkčnosť popísanú vyššie bez štruktúry viditeľnej zvonka), systémových (napr. shell) a aplikačných programov. Oproti DOSu je výhoda v tom, že verejné sú len rozhrania pre používateľov (príkazy systému) a API volaní systému (rádovo desiatky funkcií). Ak napíšete nový UNIX, stačí sa držať týchto rozhrani – nech sa nikto nepýta, čo je vovnútri.

AIX (verzia UNIXu od IBM)

už má dvojvrstvové jadro (delené podľa funkčnosti).

Machintos

(veľmi populárny prístup) redukuje jadro na dobre definovanú množinu základných funkcií a funkčnosť presúva do systémových programov – mikrokernel.

Vrstvená štruktúra OS

je modulárna štruktúra, ktorá postupne vytvára vrstvy (najnižšie je HW, najvyššie používateľské prostredie). Vrstvy si môžeme predstaviť objektovo orientovane: vrstva má dáta a poskytuje operácie pre prácu s nimi. V implementácii využíva služieb nižšej vrstvy, ale z jej služieb zverejňuje len to, čo uzná za vhodné. Takýto systém sa samozrejme buduje zdola nahor. Problematické je samozrejme rozčlenenie na vrstvy. Napr. ovládač pre odkladací (swapovací) priestor musí byť nižšie, než správca op. pamäti, ktorý používa jeho služby.

Príklady

systémy **OS/2 a WinNT**. Znemožňujú tak priamy prístup k hardwaru – to je aj jedna z príčin menšej spätnej kompatibility
s MS-DOSom.

Windows NT

má hlavné vsrtvy:

- HAL (hardware abstraction layer – vyrovnáva napr. špecifiká

V/V rozhrania, radičov prerušení apod.),

- NT executive (jadro obsahujúce správcov zhruba podľa funkčnosti správca pamäti, procesov, súborov, vyrovnávacích pamätí, V/V zariadení, bezpečnostný monitor),
- chránené podsystémy,
- aplikačné programy.

Virtuálne stroje

Pri vrstvenej štruktúre OS každá vrstva môže nazerať na rozhranie "pod sebou" ako na počítač poskytujúci určité služby – inštrukcie. Napr. aplikačné programy môžu zavolať funkciu open na otvorenie súboru ako keby to bola primitívna operácia počítača, aj keď v skutočnosti je implementovaná nižšou vrstvou OS. Principiálne teda nemusia rozlišovať medzi takouto funkciou a napr. HW inštrukciami ("vynásob 2 čísla!") – sú to všetko využiteľné služby počítača. Takýto **abstraktný pohľad na počítač ako množinu služieb je základom myšlienky virtuálneho stroja.**

OS môže všetkým používateľským procesom vytvoriť dojem, že majú pre seba kompletný "virtuálny" počítač – CPU (máme predsa time-sharing!), pamäť (tzv. virtuálna pamäť, ktorá sa ešte bude preberať), ale aj periférie (s časovým multiplexovaním alebo vydelením oblastí diskov, ktoré sa potom tvária ako celé disky – tzv. minidisky) a operačný systém. Samozrejme sú aj problémy: musíme simulovať aj user režim a privilegovaný režim: budeme teda mať virtuálny privilegovaný režim (napr. pri obsluhe virtuálnej tlačiarne), a prechod do "ozajstného" privilegovaného režimu nastane až vtedy, keď virtuálna tlačiareň pristúpi k ozajstnej fyzickej tlačiarni. Problematická je aj komunikácia – samozrejme cez virtuálnu sieť virtuálnych strojov.

Príklady:

Prominentným príkladom tejto koncepcie je VM/CMS od IBM, ale aj emulátory obľúbených personálnych systémov na iných (emulátor MS-DOSu na Macu alebo Sune).