MTA Számitástechnikai és Automatizálási Kutató Intézete, Közlemények 18/1977.

TÁROLÓVAL VALÓ GAZDÁLKODÁS KISSZÁMOLÓGÉPEK OPERÁCIÓS RENDSZERÉBEN

Salamon Márton
MTA Központi Fizikai Kutató Intézete

A TPA 70 kisszámológép diszkes operációs rendszerének tervezése során kerestük az operációs rendszerek elméletének e feladatnál alkalmazható eredményeit. Kisszámológépeknél az általánosan jelentkező problémák egy része különös jelentőséget kap a gép viszonylag kis operativ memóriája miatt. A tervezési szempontok között egyik legfontosabb az volt, hogy a rendszer központi része, a Supervisor tegye lehetővé a rendelkezésre álló operativ memória maximális kihasználását a programok számára. A Supervisornak egyrészt ki kell elégiteni a rendszerben futó forditóprogramok (Assembler, Fortran, Basic) igényeit, másrészt lényeges, hogy célrendszerekben is alkalmazható legyen, azaz I/O rendszerét könnyen lehessen bőviteni.

Az operációs rendszer tervezésekor figyelembe kellett vennünk a hardware környezetet:

- A TPA 70 egy 16 bit szóhosszuságu, korszerü kisszámológép minimum 8K szó, maximum 32K szó memória kapacitással.
- Virtuális tároló kezelést vagy overlay szervezést támogató hardware segitség nincs.
- Az operációs rendszernek mind mozgófejes, mind fixfejes diszkkel hatékonyan kell müködnie.
- Előre nem ismert perifériák vezérlésének megoldhatóságát biztositani kell.

Az operációs rendszer vezérlő része a Supervisor. A Supervisor moduláris felépitésü. Különböző (programból kiadott vagy operátori) igények hatására más-más modul vagy modulok müködnek.

Martin az operációs rendszerek tervezéséről irt könyvében 9 tényezőt sorol fel, amelyek szük keresztmetszetet jelenthetnek egy operációs rendszer müködése során. Ezek a következők:

- Memória terület
- Központi egység idő
- Háttér tároló kapacitás
- Csatorna kihasználás
- Mágneslemezes háttértárolóknál a fejmozgatások ideje
- Adatátviteli buffer terület
- Adatátviteli vonalak kihasználása
- Terminálok kihasználsa
- Operátori beavatkozások

Az alábbiakban a memória terület kapacitást vizsgáljuk abból a szempontból, hogy a felhasználói program számára hogyan lehet minél nagyobb területet biztositani. Nem foglalkozunk azzal a kérdéssel, hogy a felhasználó hogyan tudja kihasználni a legjobban a rendelkezésre álló területet.

Kézenfekvőnek látszik, hogy a Supervisor rutinjai (moduljai) közül minél többet a háttértárolón tartsunk, és csak szükség esetén hozzuk be a memóriába. Ez az eljárás azonban

- Központi egység időt
- Háttér tároló kapacitást
- Csatorna időt
- Fejmozgatást

igényel. Látható, hogy egy rutin behozása a memóriába négy másik tényezőnél játszik szerepet, tehát nem mindig ez a takarékoskodás legkifizetődőbb módja, hiszen a rendszer hatékonysága nagymértékben csökkenhet. Éppen ezért nagyon alapos megfontolást igényel az, hogy a Supervisor rutinjait hogyan kezeljük.

A TPA 70 operációs rendszerénél a Supervisor rutinjait három csoportba osztottuk.

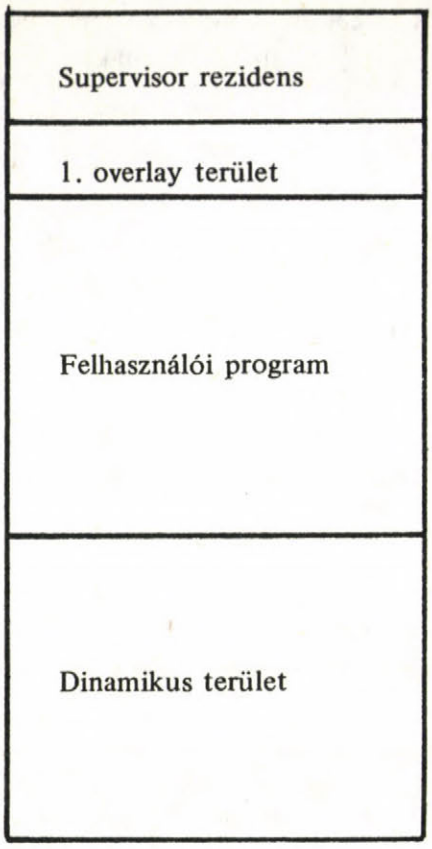
- Rezidens rutinok, amelyekre gyakran, vagy nagyon gyorsan van szükség, ezért állandóan a memóriában vannak.
- Overlay rutinok, amelyek a ritkábban előforduló igények esetén kerülnek be a memóriába.
- Perifériák handlerjei, és más, handler tipusu rutinok, amelyek a felhasználói program futása, vagy annak egy része alatt rezidenssé tehetők.

Az előzőhöz hasonló probléma a táblázatok, buffer területek, munka területek kérdése. Vannak táblázatok, amelyek állandóan szükségesek, mások csak egy-egy perifériális átvitel ideje alatt. Átviteli buffer területre is csak meghatározott ideig van szükség, de a vezérlő terminálnak állandó buffer kell.

Az előzőekben az látható, hogy a program pillanatnyi igényeivel szoros kapcsolatban van a Supervisor számára egy adott időben szükséges terület mérete. Például minél több perifériálás átvitel van folyamatban egyidejüleg, annál több munkatábla lefoglalása szükséges abban az időintervallumban.

Ezen okok miatt célszerü a memória egy részét dinamikusan kezelni, amely területből szükség szerint ki lehet hasitani egy darabot, illetve fel lehet szabaditani egy lefoglalt darabját,

A TPA 70 operációs rendszerénél a következő memória terület kiosztást találtuk célszerünek:



1/a. ábra

1/b. ábra

Az 1/b ábra a dinamikus terület egy lehetséges állapotát mutatjuk meg, amely az egymás után beérkező igények hatására alakulhat ki.

A Supervisor rezidens területe nem kiván bővebb magyarázatot.

Az 1. overlay területen olyan rutinok váltogatják egymást, amelyekre a program igényeitől függetlenül is szüksége lehet a Supervisornak (konverziós rutinok, rendszer paramétereit szolgáltató rutinok, stb.).

A dinamikus területből hasitódik ki a 2. overlay terület ha a program olyan Supervisor szolgáltatásokat igényel, amelyek legtöbb programnál csak ritkán fordulnak elő (pl. file létrehozása, törlése, stb.), de viszonylag nagy méretü rutinok szükségesek kielégitésükhöz.

A felhasználói program területét és a dinamikus terület elválasztó szaggatott vonalról ejtsünk néhány szót.
A felhasználói program és a dinamikus terület határát a felhasználói program határozza meg egyrészt saját méretével, másrészt igényeivel. A program igényeitől függően kerülnek a memóriába, illetve törlődnek perifériák handlerjei, foglalódnak le buffer területek, munkaterületek. Tul sok ilyen igény esetén a dinamikus területből kihasitható terület elfogy, és csak operátori

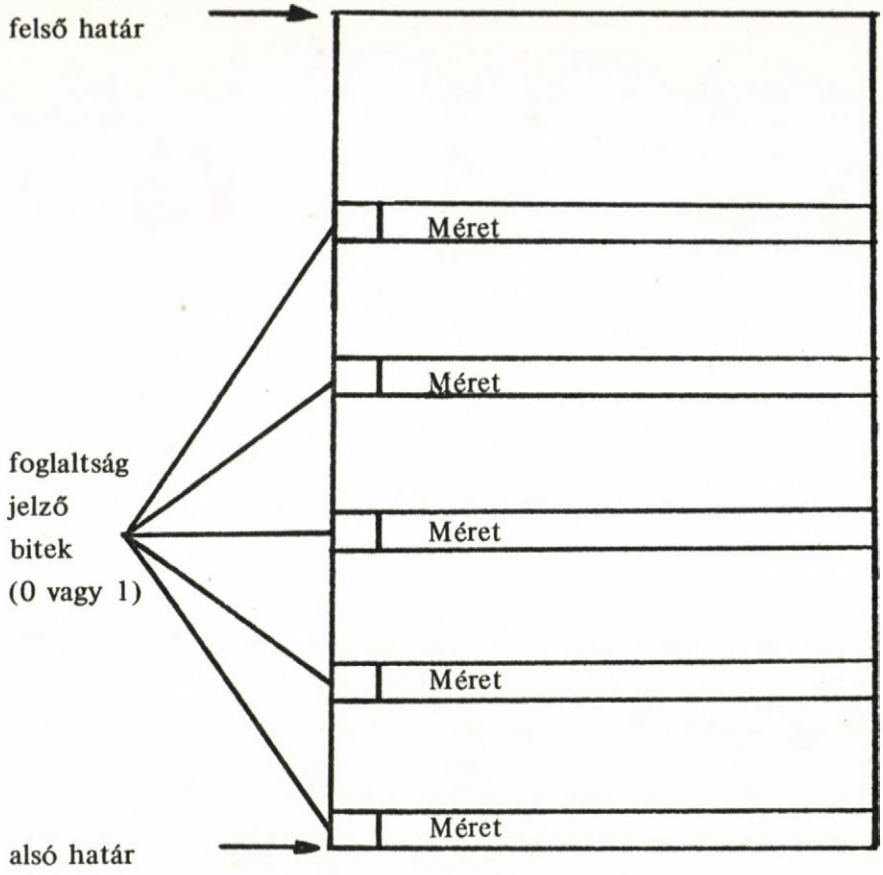
beavatkozással (pl. handler egy törlése) lehet a programot tovább futtatni. Ha ez nem megoldható akkor a program átalakitására van szükség (pl. overlay szervezés beépitése). A felhasználói program betöltésekor a dinamikus terület felső határa és a program alsó határa megegyezik. A programnak módja van ezen érték lekérdezésére, sőt szükség esetén igényelheti annak megváltoztatását is.
Ha a programozó nagy programot akar irni, akkor vigyáznia kell arra, hogy egyidejüleg csak a ténylegesen szükséges handlerek legyenek a memóriában, csak a szükséges file-ok legyenek megnyitva, és csak a szükséges bufferek legyenek lefoglalva. Ily módon növelheti saját programjának rendelkezésre álló területet a határ mozgatásával.

Fontos megjegyezni, hogy a tranziens rutinoknak és a dinamikus területre kerülő programoknak helytől független programoknak (position independent code) kell lenniük, hogy a memóriába bárhová betölthetők legyenek.

Szándékunkban van a Supervisor használata során statisztikákat vezetni arról, hogy egyes rutinokat milyen gyakran használnak. A statisztikák kiértékelése alapján módosithatjuk a rutinok csoportositását és esetleg több overlay területet vezethetünk be.
A dinamikus tároló terület kezelése:

A dinamikus tároló területet kezelő rutin memória rezidens. Mint minden ilyen rutinnál, ennél is nagyon fontos, hogy minél kisebb méretü legyen. Ez a követelmény azonban maga után vonja azt is, hogy egyszerü algoritmust kell alkalmazni.

A dinamikus területből tetszésszerinti méretü cella igényelhető, az előzetesen nincs felosztva semmiféle szempont szerint. A dinamikus tárolót kezelő rutin az alsó határtól kiindulva keres olyan összefüggő szabad területet, amely pontosan megegyezik az igényelttel. Ha éppen ekkorát nem talált, akkor a legnagyobb szabad területből hasitja ki a cellát, ilymódon elkerülhető, hogy a terület tulságosan felaprózodjon. A szükséges adminisztrációját a cellának a következő szóban végzi el. A dinamikus terület képe a következő lesz:



2. ábra

Terület felszabaditásakor csak a cella kezdőcimét kell megadni, és az szabaddá válik.

Ha már nincs elég nagy összefüggő szabad terület, akkor először az egymás mellett lévő szabad cellák összevonása történik meg, ha ez sem hozza meg a kivánt eredményt, akkor kerül csak sor a cellák elcsusztatására.
A fent leirt módon egy egyszerü, gyors rutin el tudja végezni a dinamikus tároló terület kezelését.

Az ismertetett módszernek külön előnye az, hogy akkor, ha a felhasználói programról tudjuk, hogy egyidejüleg a rendszer mely erőforrásait köti le, akkor becslés adható a feladathoz szükséges memória méretére, illetve adott memóriánál megadható a felhasználói program lehetséges maximális mérete.

S u m m a r y

Memory utilization of minicomputers

Marton Salamon

At the operating systems of minicomputers to utilize the core memory is particularly important. It is a general solutions, that beside a small resident part the transient moduls get a definite part of the memory. To decrease the number of disk transfers we must to fix in the memory certain routines (e.g. handlers) for a time period. The size of the buffers depends on the program too.

This paper describes a solution, wich takes the variable memory requirement of the user program in to account.

Р Е З Ю М Е

ИСПОЛЬЗОВАНИЕ ПАМЯТИ В МАЛЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИНАХ

М. Шаламон

В малых вычислительных машинах особенно важно экономичное использование операционной системой основной памяти. Общим решением этой проблемы является то, что рядом с маленькой резидентной частью транзиентные подпрограммы получают определенную часть памяти. С целью уменьшения вводов с диска, эти подпрограммы (например "handler") на некоторое время нужно считать резидентными. От программ пользователя зависит еще и размер буфферов.

В настоящей статье рассматривается решение, когда максимально применяется во внимание динамическая потребность в памяти программы использователя, работающей под операционной системой.