Linux sistēmas programmēšana - MD5

Algoritmu apraksti:

- **First fit** - Katru reizi rezervējot atmiņu sāk meklēt no atmiņas sākuma. Tiek izmantots pirmais atmiņas bloks, kurš ir pietiekoši liels, lai tiktu rezervēts nepieciešamais atmiņas daudzums.
- **Next fit** - meklē nākamo pieejamo bloku aiz pēdējā alocētā bloka. Ja sasniegtas atmiņas segmenta beigas, meklēšana tiek uzsākta no sākuma
- **Best fit** - meklē pēc izmēra vislabāk atbilstošo bloku (vismazākā starpība starp atmiņas bloka izmēru un atmiņas apjomu, kas jāalocē)
- **Worst fit** - meklē pēc izmēra vissliktāk atbilstošo bloku (vislielākā starpība starp atmiņas bloka izmēru un atmiņas apjomu, kas jāalocē)

Mērījumu un novertējumu apraksts

Veiktspējas mērījumi

Testa faili / Algoritms	First fit	Next fit	Best fit	Worst fit
chunks1 / requests1	13μs	8μs	14μs	14μs
chunks2 / requests2	7μs	4μs	12μs	12μs
chunks3 / requests3	7μs	4μs	13μs	13μs
chunks0 / requests0	1μs	1μs	1μs	1μs
chunks4 / requests4	2μs	2μs	2μs	2μs
chunks5 / requests5	2μs	2μs	4μs	3μs
chunks6 / requests6	2050μs	14μs	4400μs	4120μs
Vidēji:	297.43μs	5.57μs	635.14μs	595μs

Pieprasītā vs rezervētā atmiņa

Testa faili / Algoritms	Pieprasītā atmiņa	First fit	Next fit	Best fit	Worst fit
chunks1 / requests1	946	741	741	741	703
chunks2 / requests2	900	841	841	784	784
chunks3 / requests3	1014	649	649	737	571
chunks0 / requests0	2	0	0	0	0
chunks4 / requests4	1023	511	511	1023	511
chunks5 / requests5	1023	1023	1023	1023	1023

chunks6 / requests6	1024	1024	1024	1024	1024
Vidēji:	847.43	684.14	684.14	761.86	659.43

Fragmentācijas mērījumi

Testa faili / Algoritms	Sākotnējā fragmentācija	First fit	Next fit	Best fit	Worst fit
chunks1 / requests1	60%	69.8%	69.8%	68.9%	75.0%
chunks2 / requests2	70.8%	87.4%	87.4%	81.6%	80.3%
chunks3 / requests3	90.9%	73.8%	87.3%	64.3%	94.4%
chunks0 / requests0	0.0%	0.0%	0.0%	0.0%	0.0%
chunks4 / requests4	50.0%	50.0%	50.0%	0.0%	50.0%
chunks5 / requests5	50.0%	0.0%	0.0%	0.0%	0.0%
chunks6 / requests6	99.9%	0.0%	0.0%	0.0%	0.0%
Vidēji:	60.23%	40.14%	42.07%	30.69%	42.81%

Secinājumi

Pēc testu rezultātiem var secināt, ka nav viena tāda atmiņas rezervācijas algoritma, kas noderētu pilnīgi visām situācijām.
NextFit algoritma stiprā puse ir tā ātrdarbība, bet tas cieš salīdzinoši lielās fragmentācijas dēļ.
chunks6/requests6 testpiemērs (1024 viena baita segmenti un tikpat daudz pieprasījumu) vislabāk parāda šī algoritma stipro pusi - tā ātrdarbību pie lielāka atmiņas bloku skaita.
Ar BestFit algoritmu ir iespējams sasniegt ļoti zemu fragmentācijas līmeni, bet tas salīdzinoši ir ļoti lēns. Katrā alokācijā apstaigājot visus atmiņas apgabalus tiek patērēts apjomīgs laika daudzums. Secinam, ka BestFit algoritms vislabāk derīgs, ja ir kritiski svarīgi alocēt pēc iespējas vairāk atmiņas.
WorstFit, kā nosaukums paredz, ar neko īpaši neizceļas, jo ātrdarbība ir gandrīz tik pat slikta kā BestFit, bet visas pārējās īpašības ir viduvējas, ja ne pat sliktākas (gan fragmentācijas, gan ātrdarbības un alocētās atmiņas rādītāji ir vieni no sliktākajiem salīdzinājumā ar citiem algoritmiem).
Secinam, ka FirstFit ir ļoti viduvējs salīdzinājumā ar pārējiem algoritmiem. Ātrdarbība nav izcila, alocētās atmiņas daudzums ir identisks ar NextFit, tomēr fragmentācijā novērojams uzlabojums pret NextFit.

Autori: Ģirts Rudzišs (gr16014) - 34%, Emīls Ozoliņš (eo16013) - 33%, Sandis Zutiņš (sz16015) - 33%