

OSM, G3

Anders Iversen
Mikkel Mastek
Anders Bjerg Pedersen

26. februar 2008

G3.1 Benchmark af systemkaldet 'getpid'

Vi sætter i denne opgave systemkaldet `getpid()` i en for-løkke. Før og efter for-løkken kaldes `gettimeofday()`, og bagefter beregnes ud fra disse start- og sluttider den samlede og gennemsnitlige kørselstid. Resultaterne udskrives til skærmen. Veed at prøve os frem, fandt vi ud af, at vi skulle op på ca. 200.000.000 systemkald, før vi fik en samlet kørselstid på over 1 sekund. Output af programmet (3 gange) ses herunder.

Forskellene i måletiderne kan skyldes, at programmet er kørt på en multi-user maskine, hvorfor det derfor vil være ganske uforudsigeligt, hvornår der er mere eller mindre belastning på processorerne.

```
bach-0 > ./getpid-benchmark
Antal systemkald: 200000000
Starttid: sec=1204032697 + usec=327829
Sluttid: sec=1204032698 + usec=547459
Samlet måletid: sec=1 + usec=219630
Gennemsnitlig måletid: usec=6.098150e-03
bach-0 > ./getpid-benchmark
Antal systemkald: 200000000
Starttid: sec=1204032699 + usec=569489
Sluttid: sec=1204032700 + usec=694258
Samlet måletid: sec=1 + usec=124769
Gennemsnitlig måletid: usec=5.623845e-03
bach-0 > ./getpid-benchmark
Antal systemkald: 200000000
Starttid: sec=1204032703 + usec=541876
Sluttid: sec=1204032704 + usec=706762
Samlet måletid: sec=1 + usec=164886
Gennemsnitlig måletid: usec=5.824430e-03
bach-0 >
```

G3.2 Benchmark af systemkaldet 'user_getpid'

Det meste af koden fra opgave 1 er her blot kopieret ind i `user-getpid-benchmark.c.`, som igen er kørt 3 gange, nu med funktionen `user_getpid()` i stedet for `getpid()`. Det

ses af outputtet, at disse kald kører væsentligt langsommere end i G3.1. Dette skyldes, at `getpid()`-systemkaldet ikke kræver et kontekstskifte fra user- til kernelmode (i hvert fald på Linux). Dette er dog måden, hvorpå `user_getpid()`-funktionen er implementeret i vores udleverede kerne. Her bruges funktionen `swapcontext()` to gange under kaldet af `user_getpid()`.

```
bach-0 > ./user-getpid-benchmark
Antal systemkald: 1000000
Starttid: sec=1204035906 + usec=639788
Sluttid: sec=1204035908 + usec=156171
Samlet måletid: sec=1 + usec=483617
Gennemsnitlig måletid: usec=1.483617e+00
bach-0 > ./user-getpid-benchmark
Antal systemkald: 1000000
Starttid: sec=1204035909 + usec=829226
Sluttid: sec=1204035911 + usec=344176
Samlet måletid: sec=1 + usec=485050
Gennemsnitlig måletid: usec=1.485050e+00
bach-0 > ./user-getpid-benchmark
Antal systemkald: 1000000
Starttid: sec=1204035913 + usec=658651
Sluttid: sec=1204035915 + usec=175078
Samlet måletid: sec=1 + usec=483573
Gennemsnitlig måletid: usec=1.483573e+00
bach-0 >
```

G3.3 Implementation af systemkald

Vi gennemgår kort, hvad der er tilføjet i de enkelte filer og hvorfor:

syscall.h:

Tilføjet `#define SYSCALL_GETTIMEOFDAY 2` for at kunne kalde vores nye systemkald fra `user.c` og give det et unikt ID.

errnum.h:

Tilføjet `#define SYSCALL_ERROR_PERM -4` for korrekt at kunne fange systemkaldet `gettimeofday()`'s returfejl `EPERM`.

kernel.c:

Tilføjet en case i `syscall_handler()` som ud fra hvad `user.c` har gemt i register `r0` udfører det korrekte systemkald (`syscall_gettimeofday()`). Desuden har vi tilføjet selve funktionen `syscall_gettimeofday()`, der anvender systemets `gettimeofday()`-funktion. Se nærmere kommentarer i kildekoden til `kernel.c`.

user.c:

Tilføjet funktionen `user_gettimeofday()`. Det bemærkes, at der i `user.h` er lagt op til, at der kun skal sendes ét argument med funktionen. Derfor smider kernen selv et ekstra `NULL` på, når `gettimeofday()` kaldes i `kernel.c`.

Mht. testen har vi blot kopieret den udleverede test og suppleret med kald til vore egen implementerede udgave af `user_gettimeofday()`. Først testes de to funktioner

med gyldige referencer til et `timeval`-element lige efter hinanden. Af outputtet ses, at vores funktion virker og giver os et resultat, der ligger meget tæt op af det indbyggede kald. I anden runde testes begge funktioner med en ulovlig reference, hvor vores egen funktion fanger denne ulovlige adresse korrekt. Output ses nedenfor:

```
app-3 > ./user-gettimeofday-test
user_gettimeofday: sec=1204045183 + usec=60833
gettimeofday: sec=1204045183 + usec=60926
user_gettimeofday: bad address!
gettimeofday: bad address!
app-3 > ./user-gettimeofday-test
user_gettimeofday: sec=1204045295 + usec=407468
gettimeofday: sec=1204045295 + usec=407539
user_gettimeofday: bad address!
gettimeofday: bad address!
app-3 >
```