

Security Engineering

4. Übung

Aufgabe 1 (fork-exec)

Entwickeln Sie ein C-Programm *start*, das beim Aufruf

```
start prog arg1 arg2 arg3 ...
```

zunächst ein *fork()* Aufruf ausführt und dann im Sohnprozeß das Programm *prog* via *execvp* mit den angegebenen Argumenten startet. Das Programm *prog* soll mit niedrigster Priorität ausgestattet werden, siehe Systemcall *setpriority()*

Der Vaterprozess soll weiterhin folgendes tun:

- Ausgabe der PID des gestarteten Prozesses *prog*,
- Ausgabe des Return-Codes von *prog* nach dessen Beendigung,
Hinweis: Siehe Macros unter *wait(2)*
- Ausgabe eines evtl. Signals (numerisch und eine Beschreibung des Signals),
das zum Abbruch von *prog* führte (siehe auch *psignal(3)*).

Die Deklaration von *main()* in *start.c* sei

```
int main(int argc, char **argv)
```

Hier ist ein richtiger Aufruf von *execvp()* dabei:

- a) `execvp(argv[1][0], argv[1])`
- b) `execvp(argc, argv)`
- c) `execvp(argv[1], argv[2])`
- d) `execvp(argv[1], argv+1)`
- e) `execvp(*argv[1], *argv[1])`
- f) `execvp(**argv, **argv[1])`
- g) `execvp(argv[1], argv[1])`

Aufgabe 2 (Semaphoren und Shared Memory)

Implementieren Sie folgendes Erzeuger–Verbraucher–Schema mit Hilfe von Shared Memory und Semaphoren. Ein Vaterprozess P_1 legt Semaphoren und Shared Memory Segment an. Danach erzeugt er den Sohnprozess P_2 , der als Prozesskopie die Semaphoren-ID und die Shared-Mem-ID kennt. Hierfür ist nur `fork()` ohne `exec*()` nötig. Der Prozess P_1 wird Daten in den gemeinsamen Speicher schreiben, die Prozess P_2 dort herauslesen wird.

Der Erzeugerprozess P_1 hält ein Array gefüllt mit `int`-Daten, deren Anzahl sei durch eine

```
#define N_DATA 2000000
```

Direktive festgelegt. Die Daten werden von P_1 zufällig erzeugt (siehe `srand48()`, `lrand48()`).

Der Verbraucherprozess P_2 soll diese Daten erhalten, indem diese über einen von P_1 und P_2 genutzten shared memory Block übertragen werden. Im shared-memory Bereich finden weniger als `N_DATA` viele Zahlen Platz, etwa

```
#define N_SHARED 2000
```

Prozess P_1 muss also die größere Anzahl Daten in mehreren Durchläufen durch den kleineren Shared-Puffer übertragen.

Hinweise: es empfiehlt sich ein schrittweises Vorgehen

- zunächst eine Lösung ohne Semaphoren und nur einen Schreib-/Lesevorgang im Shared-Memory-Bereich:
 P_2 wartet mittels `sleep()`, damit P_1 Zeit hat, die Daten zu schreiben
- danach Semaphoren hinzunehmen, es werden zwei Semaphoren benötigt:
 - eine Semaphore S_1 , mit der der Erzeuger den Lesevorgang für den Verbraucher freigibt (V-Operation)
 - eine Semaphore S_2 , mit der der Verbraucher den Schreibvorgang für den Erzeuger freigibt (V-Operation)
 - der Verbraucher muss mit einer P-Operation auf S_1 den Lesezugriff anfordern
 - der Erzeuger muss mit einer P-Operation auf S_2 den Schreibzugriff anfordern
 - initial muss Schreiben erlaubt und Lesen verboten sein
- danach mehrere Schreib-/Lesevorgänge
- beachten Sie, dass Semaphoren und Shared Memory permanente Objekte sind, die explizit gelöscht werden müssen (`semctl()`, `shmctl()`)

Aufgabe 3 (Password-Cracking (Dictionary-Attack))

Schreiben Sie ein einfaches Password-Cracking-Tool.

Die Wörterbuch-Attacke (Dictionary-Attack) bildet den verschlüsselten (oder gehash-ten) `passwd`-Eintrag für alle Wörter des Wörterbuchs. Wird dabei eine Übereinstimmung gefunden, so ist das Klartextpassword bekannt.

Beispiel: bei einem bekannten `passwd`-Eintrag von

```
$1$ZYXWVUTS$xN7BRlVyRwgiKy.Gfnss0/
```

würde die Wörterbuch-Attacke bei Benutzung der Wörter aus `/usr/share/dict/words` diese testen, z.B.

```
A      $1$ZYXWVUTS$y6t.HbZrqRxPSI5PT1eoW1
...
secpar $1$ZYXWVUTS$Px290T6yKgy4P4C511yAd0
secque $1$ZYXWVUTS$2h2opR3NHAm9NUqEGmHbE/
secre  $1$ZYXWVUTS$3kghTnCpC90405tH2RInH1
secrecy $1$ZYXWVUTS$oiE5.nXUXQ/sF.Y.M5PlI0
secret $1$ZYXWVUTS$xN7BRlVyRwgiKy.Gfnss0/
```

und bei `secret` merken, dass der gesuchte Wert gefunden wurde, somit auf das Passwort `secret` schließen.

Schreiben Sie ein Shellskript, das ein Kommando der Form

```
openssl passwd -1 -salt ...
```

benutzt, um die Passwörter folgender Benutzer zu finden, diese stammen alle aus `/usr/share/dict/words`:

```
Angela.Merkel $1$07v0C2lZ$E8qLtddshWTpEF9EJ8Nmi1
Olaf.Scholz $1$hVEumPED$w6afIaNNrvKfGFSmCcHT/
Heiko.Maas $1$XJwb3vQN$PRjRzJVTB1PWIpw1M8uq0/
Peter.Altmaier $1$xkoLaWW0$bJib5w2vV9CClcSfs0jl20
Horst.Seehofer $1$0ngrMRaj$4mXDUbpsVQgyGpsRxiSy/1
Katarina.Barley $1$Cl5hww4i$01/df84sL0w83GqJfbT.A1
Hubertus.Heil $1$.0FMxzzD$yhGV2j9.sUpYEzHKPsoNd/
Franziska.Giffey $1$Cx3h/d.G$PxUomtQg0/n9T0eGVweCH1
Andreas.Scheuer $1$S2QJ/DwC$1CWqAcPIUpBKbyR4S8o/W1
Julia.Klöckner $1$eAnz2eeh$Bzd4kfKMZkIvZ1KTTudQb1
Ursula.von-der-Leyen $1$1aaPttrp$I2Ap/roBfRoWfx3BWRzZq.
Jens.Spahn $1$tMznqlt9$7ieEwjFr5tuMU4CXX2SFX0
Svenja.Schulze $1$C1rJw.uq$1XBONI7qTNh1Dhb5RBipv1
Anja.Karliczek $1$UL8CSHra$KRplQCo26he0xlqu3UuGR1
Gerd.Müller $1$1tX9YS4X$eNhKCEpH430z4SVYewY2J/
```