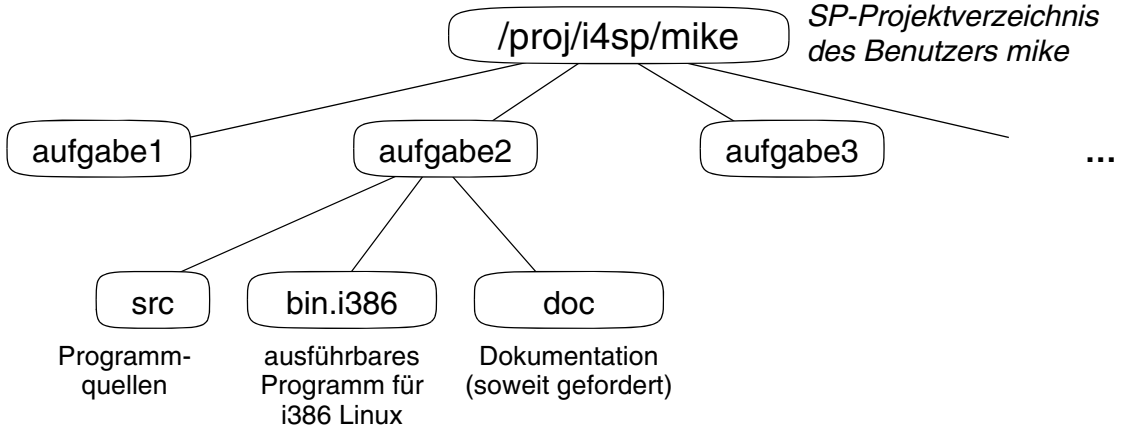
Allgemeine Hinweise zu den SP-Übungen

- Die Aufgaben sind teilweise in Zweier-Gruppen, teilweise alleine zu bearbeiten (jeweils in der Aufgabenstellung angegeben). Bei Gruppenarbeit ist der Lösungsweg und die Programmierung gemeinsam zu erarbeiten. Bei einzelnen Aufgaben wird auch eine gezielte Arbeitsteilung mit Definition von Schnittstellen gefordert.
- Die Gruppenmitglieder müssen gemeinsam an der gleichen Tafelübung teilnehmen. Die Lösung wird jeweils komplett bewertet und den Gruppenmitgliedern gleichermaßen angerechnet.
- Die Übungsaufgaben müssen spätestens bis zum jeweiligen Abgabetermin abgegeben werden. In darauffolgenden Tafelübungen werden einzelne abgegebene Lösungen besprochen — jeder Übungsteilnehmer muss dabei in der Lage sein, die gesamte Lösung seiner/ihrer Gruppe zu erläutern. Bei dieser Besprechung ist es wichtig, den Lösungsweg — z. B. mit Hilfe von Struktogrammen — und die gewählten Datenstrukturen und Betriebssystemfunktionen so zu erläutern, dass der Lösungsweg für die anderen Übungsteilnehmer nachvollziehbar wird. Es soll auf keinen Fall einfach nur Programmcode an die Tafel geschrieben werden!
- Kann jemand seine Lösung auf Anforderung nicht erläutern, wird für sie/ihn die Aufgabe als nicht abgegeben bewertet (im Zweifelsfall kann hierzu ein Gespräch außerhalb der Tafelübung stattfinden). Die abgegebenen Programme werden automatisch auf Ähnlichkeit mit anderen Programmen desselben Semesters und früherer Semester überprüft. Werden hierbei starke Übereinstimmungen festgestellt, wird die Aufgabe ebenfalls als nicht abgegeben bewertet.
- Kann ein Übungstermin nicht wahrgenommen werden, kann vorher mit den Übungsleitern ein Ersatztermin in einer anderen Übungsgruppe vereinbart werden.
- Jeder Benutzer erhält für SP ein spezielles Projektdirectory mit dem Namen /proj/i4sp/, gefolgt von dem eigenen Login-Namen. Die Projektverzeichnisse werden für alle im WAFFEL zu einer Tafelübung angemeldeten Teilnehmer nächtlich erstellt. Eine Anmeldung ist zur Übungsteilnahme daher **zwingend**. Der Verzeichnis-Baum für die Aufgaben ist folgendermaßen aufzubauen:



- Die Teilbäume mit den Übungsaufgaben dürfen nicht für andere Benutzer zugreifbar sein. Die Zugriffsrechte werden automatisch überprüft, bei falschen Zugriffsrechten wird die Abgabe nicht gewertet.
- Die Aufgaben sind bis spätestens zum Abgabetermin durch Aufruf des Programms

  /proj/i4sp/pub/abgabe aufgabe***X*** ***X*** =1 ... 9

  abzugeben. Dieses Programm überprüft die Verzeichnisstruktur und die Namen der Dateien, die nach der Aufgabenstellung vorhanden sein müssen, und erzeugt dann ein Archiv mit den abgegebenen Dateien. Bis zum Abgabetermin kann ein Programm beliebig oft abgegeben werden — es gilt der letzte, vor dem Abgabetermin vorgenommen Aufruf des Abgabeprogramms.
- Sollten Sie aus triftigem Grund eine fristgerechte Abgabe versäumen, so ist eine verspätete Abgabe durch Aufruf des Abgabeprogramms mit der Zusatzoption **-force** möglich:

  /proj/i4sp/pub/abgabe -force aufgabe***X*** ***X*** =1 ... 9

  Damit eine solche Abgabe bei der Wertung berücksichtigt wird, ist jedoch in jedem Fall eine Rücksprache mit Ihrem Übungsleiter erforderlich.

Aufgabe 1:

24.10.2008

slist (6 Punkte)

Implementieren Sie eine verkettete Liste, welche positive Ganzzahlen aufsteigend sortiert verwaltet. Auf die Liste soll mit den folgenden Funktionen zugegriffen werden:

- `int insertElement(int value)`: Fügt einen Wert an die richtige Position in der Liste ein, wenn dieser noch nicht vorhanden ist. Im Erfolgsfall gibt die Funktion den Wert zurück, ansonsten einen negativen Wert.
- `int removeElement(int value)`: Entnimmt den angegebenen Wert aus der Liste und gibt diesen zurück. Ist der Wert in der Liste nicht vorhanden, wird ein negatives Ergebnis zurückgeliefert.
- `int getMinVal(), int getMaxVal()`: Liefern das Minimum bzw. Maximum der in der Liste enthaltenen Werte zurück. Ist die Liste leer, wird ein negativer Wert zurückgeliefert.
- `void printList()`: Gibt die Liste auf die Standardausgabe aus. Ist die Liste leer, wird *(empty)* ausgegeben. Die Ausgabe wird mit einem Zeilenumbruch abgeschlossen.

Das Hauptprogramm soll einige Werte in die Liste einfügen und wieder entnehmen. Die folgende Beispiel-Codesequenz

```
printf("insert 47: %d\n", insertElement(47));
printf("insert 11: %d\n", insertElement(11));
printf("insert 23: %d\n", insertElement(23));
printf("insert 11: %d\n", insertElement(11));
printList();
printf("Min: %d, Max: %d\n", getMinVal(), getMaxVal());
printf("remove 11: %d\n", removeElement(11));
printf("remove 11: %d\n", removeElement(11));
printList();
```

soll folgende Ausgabe erzeugen:

```
insert 47: 47
insert 11: 11
insert 23: 23
insert 11: -1
11 -> 23 -> 47
Min: 11, Max: 47
remove 11: 11
remove 11: -1
23 -> 47
```

Hinweise:

Das C-Programm ist in der Datei **slist.c** im **src**-Verzeichnis abzulegen. Das Programm muss dem ANSI-C und POSIX Standard entsprechen und mit dem GNU-C-Compiler auf den Linux-Rechnern im CIP-Pool compilieren. Dazu ist der Compiler mit folgenden Parametern aufzurufen.

gcc -ansi -pedantic -D_POSIX_SOURCE -Wall -Werror -o slist slist.c

Unterprogramme und globale Variablendefinitionen sind ausreichend zu kommentieren. Achten Sie bitte außerdem auf saubere Gliederung des Quellcodes!

Wir erwarten, dass diese Aufgabe in **Zweier-Gruppen** bearbeitet wird.

Abgabe: bis spätestens Donnerstag, 06.11.2008, 14:00 Uhr